Haaga-Helia
ammattikorkeakoulu Oy

Avoimen lähdekoodin toiminnanohjausjärjestelmät Axelor, Metasfresh ja Tryton

Jaakko Vehkamäki

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2020

Haaga-Helia
ammattikorkeakoulu Oy

Tekijä(t)
Jaakko Vehkamäki

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma, päivä

Raportin/Opinnäytetyön nimi
Avoimen lähdekoodin toiminnanohjausjärjestelmät Axelor, Metasfresh ja Tryton

Sivu- ja liitesivumäärä
36 + 18

Tämän opinnäytetyön tarkoituksena oli selvittää avoimen lähdekoodin toiminnanohjausjärjestelmiä Axelor, Metasfresh ja Tryton tutkien, voidaanko niitä nähdä potentiaalisena vaihtoehtona kaupallisille suljetun lähdekoodin järjestelmille pienen ja keskisuuren yrityksen käytössä.

Työn teoriataustassa paneudutaan toiminnanohjausjärjestelmistä ja niiden käytettävyydestä tehtyihin tutkimuksiin ja käydään läpi keskeisiä käsitteitä toiminnanohjausjärjestelmien, avoimen lähdekoodin, lisenssien ja aihepiiriin liittyvien prosessien osalta.

Tutkimus perustuu kvalitatiiviseen tutkimustapaan ja se on toteutettu asentamalla tutkitut toiminnanohjausjärjestelmät tutkimusta varten luoduille virtuaalipalvelimille. Asennus- ja testausvaiheen jälkeen niiden etenemisestä on tehty teoriataustaan perustuva analyysi asennettavuutta ja käytettävyyttä painottaen. Asennusvaihetta tarkasteltiin sen helppoutta, sen vaatimaa aikaa ja kustomoitavuutta arvioiden. Käytettävyyttä tarkasteltiin kahdesta eri näkökulmasta, jotka olivat peruskäytettävyys ja ulkoasu, sekä saatavilla oleva käyttäjätuki ja ohjeet.

Tutkittujen toiminnanohjausjärjestelmien asennus- ja testausvaiheen aikana tehdyt havainnot kirjattiin ylös ja niitä analysoitiin tutkimuksessa teoriataustaan perustuen.

Tehtyjen havaintojen ja teoriataustan pohjalta voitiin todeta, että avoimen lähdekoodin toiminnanohjausjärjestelmät tarjoavat parhaimmillaan kaupallisia suljetun lähdekoodin järjestelmiä vastaavan laadun. Lisäksi havaittiin, että avoimen lähdekoodin toiminnanohjausjärjestelmän käyttöönottoa suunniteltaessa on mahdollisen järjestelmätoimittajan tarjoaman asiantuntemuksen puuttumisen vuoksi suositeltavaa kiinnittää erityistä huomiota käyttöönottoprojektiin ja sen suunnitteluun.

Avoimen lähdekoodin toiminnanohjausjärjestelmän käyttöönottoprojektin merkittävimpinä haasteina voidaan tämän tutkimuksen perusteella pitää riittävän käyttöohjeistuksen järjestämistä kielellä, jonka kaikki järjestelmän käyttäjät hallitsevat ja laadukkaan käyttäjäkoulutuksen järjestämistä.

Asiasanat
ERP, Tietojärjestelmät, Toiminnanohjausjärjestelmä, Tietohallinto, Avoin lähdekoodi, Open source

Sisällys

1 Johdanto

Toiminnanohjausjärjestelmät ovat viimeisten vuosikymmenien aikana tulleet osaksi monen hiemankaan suuremmassa mittakaavassa toimivan yrityksen ja yhteisön arkea. Parhaimmillaan ne helpottavat suuresti hallinnoimaan toiminnan kannalta välttämättömiä tietoja, aina asiakasrekisteristä kirjanpidon tapahtumien ja lukujen dokumentointiin.

Jossakin vaiheessa tulee väistämättä tarve uuden toiminnanohjausjärjestelmän käyttöönotolle tai vanhan käytössä olevan päivittämiselle. Suomesta ja muualta maailmasta löytyy useita toimijoita, jotka tarjoavat eri valmistajien suunnittelemia kaupallisia suljetun lähdekoodin toiminnanohjausjärjestelmiä ja niiden lisäosia valmiina tuotteena tai asiakkaan tarpeiden mukaisesti räätälöitynä. Valmis toiminnanohjausjärjestelmä voidaan toimittaa joko asiakkaan omalle palvelimelle asennettuna, tai vaihtoehtoisesti pilvipalveluna järjestelmätoimittajan ylläpitämälle palvelimelle asennettuna.

Kaupallisia toiminnanohjausjärjestelmiä vähemmän tunnettu vaihtoehto ovat avoimen lähdekoodin toiminnanohjausjärjestelmät, joiden ero suljetun lähdekoodin toiminnanohjausjärjestelmiin on se, että niiden lähdekoodi on julkaistu vapaasti kaikkien saataville ja riippuen lisenssistä, jonka alla järjestelmä on julkaistu, se voi olla joko vapaasti tai tietyin rajoituksin kaikkien kehitettävissä ja käytettävissä. Kaupallisten toiminnanohjausjärjestelmien lähdekoodi on yleensä suljettu ja oikeus sen muokkaamiseen on rajoitettu yhteistyökumppaneille.

Tässä tutkimuksessa mentiin pintaa syvemmälle ja pyrittiin selvittämään, onko avoimen lähdekoodin toiminnanohjausjärjestelmä varteenotettava vaihtoehto suljetun lähdekoodin toiminnanohjausjärjestelmälle pienen ja keskisuuren yrityksen käytön kannalta. Tutkimus toteutettiin asentamalla kolme avoimen lähdekoodin toiminnanohjausjärjestelmää virtuaalipalvelimelle ja arvioimalla niiden asennettavuutta, käytettävyyttä sekä saatavilla olevaa käyttäjätukea ja ohjeita. Käytettävyysarviointi perustui purchase to pay ja order to cash -prosesseihin liittyvien toimintojen suorittamiseen.

Tutkimus rajautuu kolmeen avoimen lähdekoodin toiminnanohjausjärjestelmään, jotka ovat Axelor, Metasfresh ja Tryton. Tutkittavien järjestelmien valinnassa kiinnitettiin huomiota siihen, että ne täyttävät GDPR -asetuksen (Europa 2020.) asettamat vaatimukset. Kaikki muut toiminnanohjausjärjestelmät jäävät tutkimuksen ulkopuolelle. Tässä tutkimuksessa ei myöskään huomioitu tutkittaviin järjestelmiin tarjottavia kolmansien osapuolien lisäosia tai palveluita. Tutkimuksessa lähdettiin siitä, että toiminnanohjausjärjestelmä otetaan käyttöön käyttäjän omalle palvelimelle käyttäjän omin resurssein.

2 Tavoite ja rajaus

Tutkimuksen tavoitteena oli tutkia kolmea avoimen lähdekoodin toiminnanohjausjärjestelmää verraten, kuinka potentiaalisina vaihtoehtoina niitä voidaan asennettavuutensa ja käytettävyytensä kannalta pitää suljetun lähdekoodin toiminnanohjausjärjestelmälle.

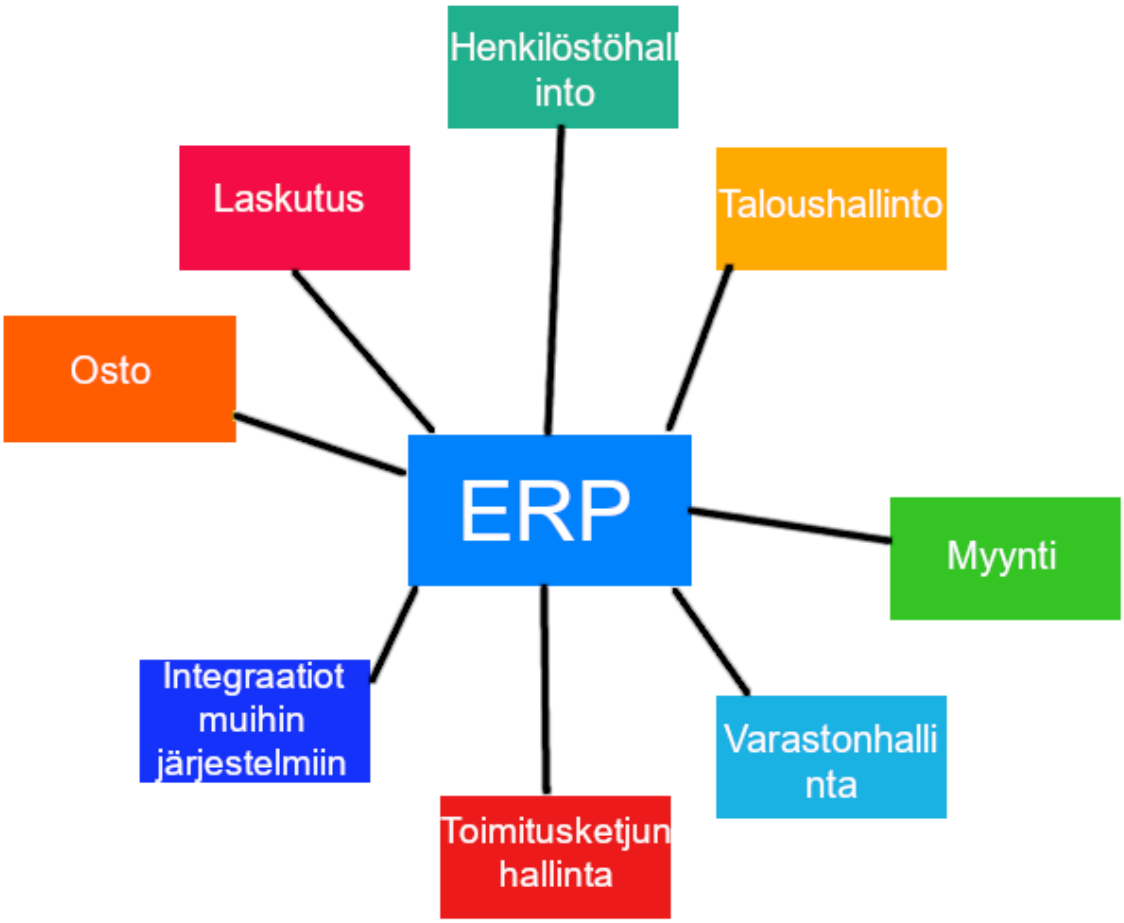
Tutkimus keskittyi Axelor, Metasfresh ja Tryton -nimisiin toiminnanohjausjärjestelmiin, jättäen ulkopuolelle kaikki muut tarjolla olevat järjestelmät. Kyseisiin järjestelmiin mahdollisesti tarjolla olevia kolmansien osapuolien lisäosia ei tässä tutkimuksessa huomioitu. Myöskään järjestelmien kehittäjien mahdollisesti tarjoamia maksullisia lisäosia tai -palveluita ei sisällytetty tähän tutkimukseen. Tutkimuksessa käytetyt palvelimet ja niiden hallinta jäivät tutkimuksen ulkopuolelle siltä osin, kun se ei liity suoraan toiminnanohjausjärjestelmän asentamiseen.

Tutkimus tehtiin ensisijaisesti pienten ja keskisuurten yritysten toimintaa ajatellen, mikä tarkoittaa yrityksiä, joissa työskentelee alle 250 henkilöä (Yrittäjät 2020).

3 Teoreettinen viitekehys

Avoimen lähdekoodin toiminnanohjausjärjestelmistä ja niiden soveltuvuudesta yrityskäyttöön on tehty jonkin verran aiempia tutkimuksia sekä Suomessa, että ulkomailla. Tämän tutkimuksen taustamateriaalina on käytetty useita toiminnanohjausjärjestelmistä ja käytettävyydestä tehtyjä tutkimuksia. Toiminnanohjausjärjestelmien käyttöönottoa ja käytettävyyttä on myös tutkittu paljon. Tämän luvun alkuun on listattu aikaisemmissa tutkimuksissa tehtyjä havaintoja sekä avoimen lähdekoodin toiminnanohjausjärjestelmistä, että toiminnanohjausjärjestelmistä, niiden käyttöönotosta ja käytettävyydestä yleisesti.

3.1 Toiminnanohjausjärjestelmä

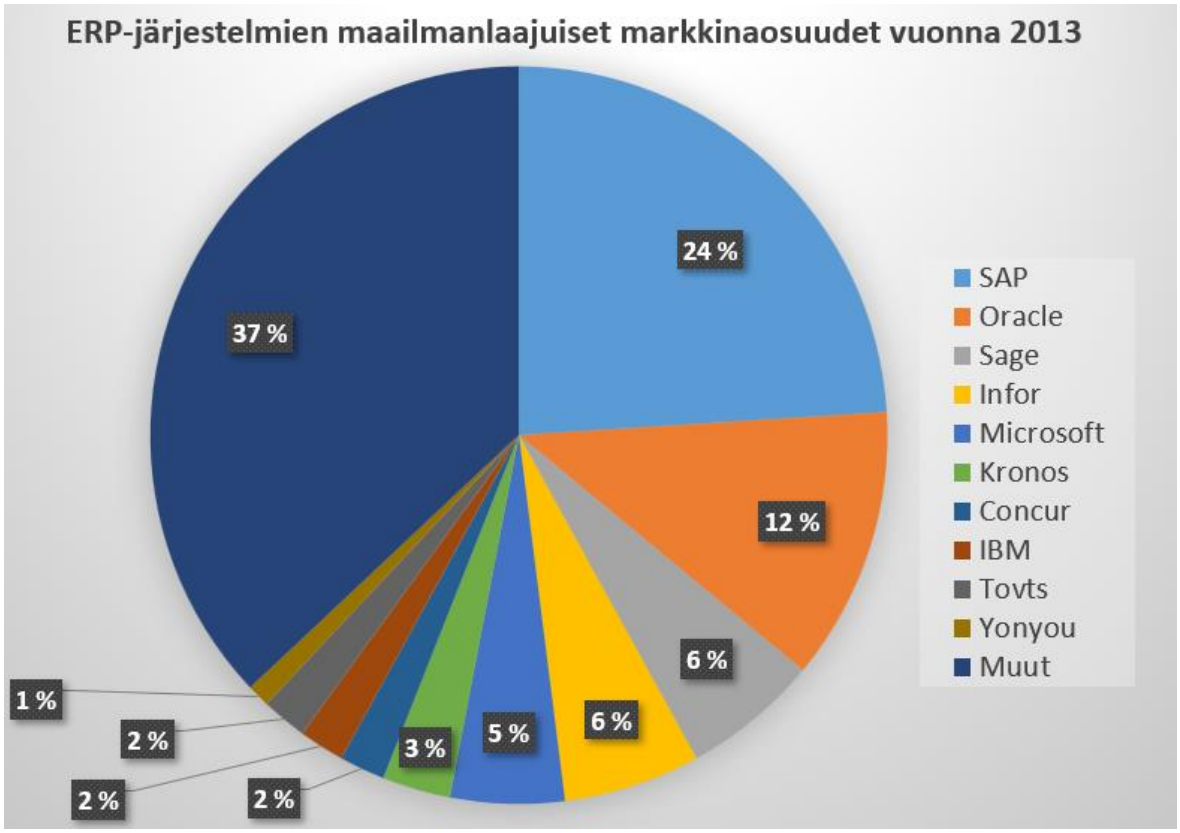


Kuva 1. Toiminnanohjausjärjestelmä ja sen yleisiä toiminnallisuuksia, eli moduuleja (mukaillen Datavtech 2019)

Toiminnanohjausjärjestelmä, josta käytetään yleisesti myös englannin kielen sanoista Enterprise Resource Planning tulevaa lyhennettä ”ERP-järjestelmä”, on yrityksen tietojen ja resurssien hallinnointia varten kehitetty tietojärjestelmä. Toiminnanohjausjärjestelmä voi yrityskohtaisten tarpeiden mukaan koostua useista eri osista, joista tavanomaisia ovat esimerkiksi kirjanpito, laskutus, myynti, osto, varastonhallinta ja asiakkuuksienhallinta.

Kuvassa 1 on kuvattu joitakin toiminnanohjausjärjestelmän yleisimmistä osista. (Investopedia 2019.)

Toiminnanohjausjärjestelmien historia alkaa 1960-luvulta, jolloin alettiin kehittämään ja ottamaan käyttöön yksinkertaisia varastonhallintajärjestelmiä. Seuraavien vuosikymmenien aikana järjestelmät kehittyivät kattamaan myös esimerkiksi taloushallinnon ja henkilöstöhallinnon. ERP-järjestelmät tulivat 1990-luvulla, jonka aikana niihin kehitettiin lisää moduuleja ja toiminnallisuuksia. (Hossain, L, Patrick, J & Rashid, M. 2002. 4-5).



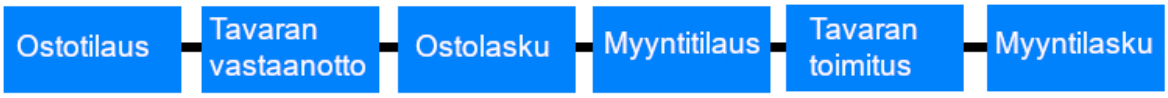
Kuva 2. Maailmanlaajuiset markkinaosuudet vuonna 2013 (mukaillen Tadviser 2020)

Kuvan 2 kaavio osoittaa, että kaupallinen toiminnanohjausjärjestelmä SAP oli vuonna 2013 markkinoiden suurin toimija lähes neljänneksen markkinaosuudella. Hieman yli 50% maailmanmarkkinoista jakautui viiden kaupallisen järjestelmän kesken, joilla oli kaikilla vähintään 5% markkinaosuus. (Tadviser 2020.)

3.2 Purchase to pay ja order to cash -prosessit

Järjestelmiä testattiin niiden käytettävyyden osalta purchase to pay ja order to cash -prosesseista tuttujen työvaiheiden avulla. Näiden prosessien yleisesti tunnetusta kulusta poiketen maksuliikenteen osuus jätettiin tässä tutkimuksessa pois. Kaikkiin järjestelmiin luotiin prosesseja varten tarvittavat kumppanit ja nimikkeet.

Purchase to pay -prosessi tarkoittaa nimensä mukaisesti tapahtumaketjua ostotilauksen ja tilaukseen liittyvän ostolaskun maksamisen välillä ja order to cash -prosessi taas tarkoittaa vastaavaa tapahtumaketjua myynnin osalta, jolloin tavarat ja maksusuoritukset liikkuvat toiseen suuntaan. (Investopedia 2020b;Techtarget 2020.)



Kuva 3. Järjestelmiin luotu tapahtumaketju vaiheittain (mukaillen Investopedia 2020b; Techtarget 2020)

Järjestelmiin luotiin niiden testausvaiheessa kuvan 3 mukaiset kuusi tapahtumaa.

3.3 Avoin lähdekoodi

Avoin lähdekoodi on ohjelmiston kehitystapa, joka antaa käyttäjälle vapauden kopioida, muokata, käyttää ja jakaa ohjelmistoa. Avoimen lähdekoodin ohjelmistojen perusversiot ovat yleensä ilmaisia. Nimensä mukaisesti avoin lähdekoodi julkaistaan kaikkien tarkasteltavaksi, mikä mahdollistaa sen vapaan muokkaamisen. Vastakohta avoimen lähdekoodin ohjelmistolle on suljetun lähdekoodin ohjelmisto, jonka lähdekoodi ei ole julkista tietoa, vaan se kuuluu monesti liikesalaisuuden piiriin. Suljetun lähdekoodin ohjelmistot ovat yleensä maksullisia ja niitä saavat muokata vain ohjelmiston kehittäjän lisensoimat tahot. (Coss 2020.)

Ohjelman lähdekoodi tarkoittaa ohjelmointikielellä kirjoitettua ohjelman perusrakennetta, jolla kerrotaan ohjelmaa ajavalle tietokoneelle ohjelman toimintaa ja eri toimintoja varten tarvittavat komennot, jotka tietokone suorittaa tietyssä järjestyksessä hakien tietoa koodissa määritellystä paikasta ja tehden koodissa määriteltyjä laskutoimenpiteitä. (Coss 2020.)

Joni Kettunen (2018, 35) on tutkinut avoimen lähdekoodin toiminnanohjausjärjestelmiä pk-yrityksissä, hän on tutkimuksessaan maininnut järjestelmätoimittajasta riippumattomuuden avoimen lähdekoodin toiminnanohjausjärjestelmien eduksi kaupallisiin suljetun lähdekoodin toiminnanohjausjärjestelmiin verrattuna. Tämä on tärkeä huomio varsinkin järjestelmän kehityksen osalta, sillä suljetun lähdekoodin järjestelmistä poiketen avoimen lähdekoodin järjestelmien kehittäminen on vapaata, mikä mahdollistaa jatkokehityksen esimerkiksi käyttäjätahon toimesta.

3.3.1 Heikkoja puolia

Avoimen lähdekoodin ohjelmiston heikkona puolena voidaan varsinkin toiminnanohjausjärjestelmän kaltaisen merkittävän tietojärjestelmän osalta pitää sitä, että sen päivittäminen ja kehittäminen voidaan kehityksestä vastaavan tahon päätöksellä lopettaa käytännössä milloin tahansa. Ilmaiseksi vapaassa jaossa olevan avoimen lähdekoodin ohjelmiston kehittäjää ei oletusarvoisesti sido minkäänlainen sopimus, joka velvoittaisi ylläpitämään ja kehittämään sitä tulevaisuudessa. (Stuart 2019.)

Avoimen lähdekoodin ohjelmistolle ei myöskään tarjota kaupallista suljetun lähdekoodin ohjelmistoa vastaavaa käyttäjätukea. Ohjelmiston kehittäjä ei ole vastuussa riittävän käyttäjätuen ja käyttöohjeiden järjestämisestä, joten käyttöohjeiden luominen ja käyttäjätuki perustuvat monesti suurilta osin käyttäjäyhteisön vapaaehtoiseen toimintaan. Joidenkin ohjelmistojen kohdalla ohjelmiston kehittäjä tai jokin kolmas osapuoli saattaa tarjota erilliseen sopimukseen perustuvaa maksullista käyttäjätukea. (Moog 2018; Stuart, G. 2019.)

3.3.2 Tietoturva

Avoimen lähdekoodin ohjelman tietoturva ja sen riskit perustuvat molemmat siihen, että koodi on julkisesti kaikkien tarkasteltavissa. Suljetun lähdekoodin ohjelman tietoturvaa parantaa se, ettei kuka tahansa pääse tarkastelemaan koodia ja etsimään siitä haavoittuvuuksia, mutta avoimen lähdekoodin ohjelmistojen kohdalla lähdekoodin julkaiseminen antaa kaikille mahdollisuuden haavoittuvuuksien etsimiseen, mikä voi parhaimmillaan nopeuttaa mahdollisten tietoturva-aukkojen löytymistä. (Leppioja & Tuomela 2017, 28-30.)

Parhaimmillaan avoimen lähdekoodin ohjelman tietoturva voi olla suljetun lähdekoodin ohjelmaa paremmalla tasolla. Markus Leppioja ja Ilari Tuomela toteavat tutkimuksessaan (2017, 30), että suljetun lähdekoodin ohjelmasta poiketen käyttäjä voi halutessaan lisätä avoimen lähdekoodin ohjelman tietoturvaa itse hankkimillaan tai kehittämillään tietoturvaratkaisuilla, koska lähdekoodin muokkaaminen on sen avoimuuden ansiosta mahdollista, kun taas suljetun lähdekoodin ohjelman kohdalla käyttäjän on luotettava ohjelman tietoturvan osalta järjestelmätoimittajaan.

3.3.3 Lisenssit yleisesti

Erilaisia avoimen lähdekoodin lisenssejä on olemassa useita. Lisenssi määrittää tarkemmin, millä ehdoin käyttäjä voi muokata, jakaa ja käyttää ohjelmaa. Lisenssien ehdoissa voidaan vaatia esimerkiksi muokattuna julkaistun avoimen lähdekoodin ohjelmaan perustuvan ohjelman lähdekoodin julkaisemista, tai ohjelman alkuperäisen tekijän mainitsemista

myös muokatussa versiossa ja lähdekoodiin tehtyjen muutosten merkitsemistä. Taulukossa 1 on listattu tutkimuksessa tutkitut järjestelmät ja niiden lisenssit. (GNU 2020b.)

Taulukko 1. Tutkimuksessa mukana olevat järjestelmät ja niiden lisenssit

Axelor	AGPLv2
Metasfresh	GPLv3
Tryton	GPLv3

GPLv2, eli GNU General Public License on vuonna 2007 voimaantullut lisenssi, jonka ehtojen mukaan ohjelmaa saa muokata, käyttää ja jakaa vapaasti, kunhan lähdekoodin muokatut osat merkitään selvästi, muokattu versio julkaistaan samalla lisenssillä ja ohjelman alkuperäinen tekijä mainitaan. (GNU 2020b.)

AGPLv2, eli GNU Affero General Public License on vuonna 2007 voimaantullut GPL-lisenssiin pohjautuva lisenssi. (GNU 2020a).

3.4 Käytettävyys

Käytettävyydellä mitataan sitä, kuinka hyvin järjestelmä tai jokin muu tuote palvelee käyttötarkoitustaan. Järjestelmän käytettävyyteen vaikuttavat käyttöliittymän lisäksi myös monet muut asiat, kuten esimerkiksi eri toimintojen aikana näytettävät ilmoitukset ja niissä käytetty kieli, sekä termit. (Smith, A. 2017.)

Hyvän käytettävyyden saavuttaminen vaatii sen, että järjestelmän käyttö on helposti opittavissa, ajankäytöllisesti tehokasta, toiminnot jäävät helposti mieleen, virheiden mahdollisuus on pyritty minimoimaan ja sen käyttäminen on mielekästä käyttäjän näkökulmasta. Edellä mainitut asiat ovat viisi yleisesti käytettävyyden mittaamiseen käytettävää laatutekijää. (Smith, A. 2017; Sippola, T. 2017; Nielsen 2012.)

Käytettävyyttä on tarkasteltu Sanna Järvelän (2011) ja Kaisu Mikkosen (2012) toiminnanohjausjärjestelmän käytettävyydestä tekemien tutkimusten havaintojen pohjalta. Molemmissa tutkimuksissa yhtenä tutkimusmenetelmänä on käytetty kyselytutkimusta ja käytettävyyden tarkastelussa on muun tietoperustan ohella huomioitu taulukossa 2 listatut Jakob Nielsenin kymmenen heuristista sääntöä. (Nielsen 1994.)

Taulukko 2. Nielsenin (Nielsen 1994) heurististen sääntöjen lista

1.	Järjestelmän tilan ja tapahtumien selkeys käyttäjälle, esimerkiksi ”toiminto suoritettu” -ilmoitusten avulla.
2.	Järjestelmän kieli ja käytetyt termit ovat käyttäjälle tuttuja.
3.	Järjestelmän hallittavuus ja mahdollisten virheiden peruuttamisen helppous.
4.	Toimintojen johdonmukaisuus ja selkeä nimeäminen, joka ei jätä tulkinnanvaraa.
5.	Virheen mahdollisuuden minimoiminen, esimerkiksi pyytämällä vahvistus ennen tärkeää toimintoa
6.	Käyttöliittymän selkeys, jotta mahdollisimman vähän jäisi pelkästään käyttäjän muistin varaan.
7.	Käyttöliittymän joustavuus, tehokkuus ja käyttäjäkohtainen muokattavuus.
8.	Käyttöliittymän esteettisyys ja minimalistisuus; toiminnoissa ei ole käyttäjän kannalta turhaa tietoa.
9.	Virheiden tunnistettavuuden ja syyn selvittämisen helppous; selkeät virheilmoitukset, jotka sisältävät virhekoodien sijasta tekstiä.
10.	Tuen ja dokumentaation helppo saatavuus, sekä selkeys.

Aiemmissa tutkimuksissa on havaittu, että Nielsenin (1994) heurististen sääntöjen listan ensimmäisessä kohdassa mainittu järjestelmän ja käyttäjän välisen kommunikaation tärkeys ilmeni käyttäjien vastauksista kyselytutkimuksessa. (Mikkonen 2012, 44-45.)

Mikko Sahanen (2014, 63) toteaa toiminnanohjausjärjestelmien käytettävyydestä tekemässään tutkimuksessa seuraavasti: ”Virhetilanteissa ongelmat toiminnanohjausjärjestelmässä saattavat johtaa pahimmillaan koko tavaran toimitusketjun katkeamiseen.” Toiminnanohjausjärjestelmä ja sen käytettävyys ovat liiketoiminnallisesti merkittäviä asioita, jotka tulee ottaa huomioon kaikissa toiminnanohjausjärjestelmiin liittyvissä projekteissa.

Sanna Järvelä (2014, 40) on käytettävyystutkimuksessaan tutkinut kohdeyrityksessä jonkin aikaa käytössä ollutta toiminnanohjausjärjestelmää. Hän on tutkimuksensa johtopäätöksissä maininnut keskeneräisen dokumentaation vaikeuttaneen ohjeistuksen hyödyntämistä. Kyselytutkimuksessa on myös tullut ilmi, että osa käyttäjistä on kokenut järjestelmän ja tarjolla olevan ohjeistuksen englanninkielisyyden ongelmallisena.

Nielsen (2012) määrittelee yhdeksi käytettävyyden viidestä laatutekijästä järjestelmän käytön oppimisen helppouden. Opittavuutta arvioitiin tässä tutkimuksessa järjestelmän perustoimintojen ja rakenteen osalta. Rakenteella tarkoitetaan tässä tapauksessa eri toimintoihin johtavia polkuja ja niiden muistettavuutta ensimmäisten käyttökertojen jälkeen.

3.5 Käyttöönotto

Toiminnanohjausjärjestelmän käyttöönotto on prosessi, joka etenee organisaation käyttöön sopivan toiminnanohjausjärjestelmän valinnasta aina järjestelmän varsinaiseen käyttöönottoon. Käyttöönottoa varten muodostetaan projektiryhmä, joka vastaa käyttöönoton suunnittelusta ja sen vaiheiden toteuttamisesta. Tavallisesti käyttöönottoprojekti kestää useampia kuukausia ja sen ylläpitovaihe jatkuu vielä järjestelmän varsinaisen käyttöönoton jälkeenkin. (Bistasolutions 2018.)

Yksi käyttöönoton vaiheista on järjestelmän asennus, asennettavuutta voidaan tarkastella arvioimalla asennuksen helppoutta, siihen kuluvaa aikaa, asennuksen vaatimaa työmäärää ja kustomoitavuutta. Lisäksi voidaan arvioida myös asennuksen poistamisen helppoutta ja toistettavuutta, mitä ei kuitenkaan tässä tutkimuksessa tehty. (Lauras, M, Zelm, M, Archiméde, B, Bénaben, F & Doumeingts, G 2015. 68; Lenhard, J. 2016 125-127.)

Toiminnanohjausjärjestelmien käyttöönottoa erilaisissa organisaatioissa on tutkittu paljon. Seuraavissa kappaleissa käsitellään joitakin keskeisiä aihepiiriin keskittyneissä tutkimuksissa tehtyjä havaintoja.

Markus Savolainen (2008, 53) kirjoittaa opinnäytetyönsä loppupäätelmässä Linux-pohjaisille palvelimille asennettavista järjestelmistä, että ”Kokonaan avoimen lähdekoodin varaan rakennettavan järjestelmän rakentamiseen tarvitaan henkilö, joka on jo ennestään tutustunut Linux-käyttöjärjestelmään. Lisäksi tarvitaan henkilö, jolle toiminnanohjausjärjestelmät ovat ennestään tuttuja.” Tämä huomio projektin vastuuhenkilöiden riittävästä järjestelmäosaamisesta on tärkeä käyttöönottoprojektin onnistumisen kannalta.

Tuukka Paananen (2017, 24) on tutkimuksessa keräämänsä ja käyttämänsä materiaalin perusteella esittänyt näkemyksen riittävän käyttäjäkoulutuksen suuresta merkityksestä projektin onnistumisen kannalta. Hän korostaa myös, että järjestelmän perustoimintojen ja eri osien käytön kouluttamisen lisäksi loppukäyttäjälle olisi tärkeää kertoa miksi asiat tehdään tietyllä tavalla, ja pyrkiä osoittamaan käyttäjälle näkyvästi uuden järjestelmän tuomat hyödyt.

Paananen (2017, 28) mainitsee tutkimuksensa yhteenvedossa yhdeksi tutkimuksen kohteena olleen järjestelmäprojektin merkittäväksi ongelmaksi järjestelmätoimittajan puutteellinen liiketoimintaosaamisen ja siitä aiheutuneet ongelmat, jotka ovat osaltaan pienentäneet uudella järjestelmällä saavutettuja hyötyjä.

Henri Teittinen (2008, 172-173) on väitöskirjassaan tutkinut ERP-kokonaisuuksia, niiden rakentumista ja taustalla vaikuttavia tekijöitä. Tutkimuksen johtopäätöksissä hän painottaa järjestelmäintegraation olevan vuorovaikutusta erilaisten maailmojen välillä. Teittisen mukaan näitä maailmoja ovat liikkeenjohto, järjestelmätoimittajat ja loppukäyttäjät. Hän huomauttaa, että on harhaanjohtavaa ajatella ERP-implementaatio ainoastaan erilaisten teknisten moduulien välisenä integraationa ja korostaa, että ERP rakentuu todellisuudessa näiden kolmen maailman välisestä integraatiosta. Teittisen kuvaaman ajattelutavan mukaisesti toimittaessa myös liiketoiminnallinen näkökulma tulee varmemmin huomioiduksi projektin aikana.

Toiminnanohjausjärjestelmän käyttöönoton ongelmiin perehtyneen tutkimuksensa yhteenvedossa Riku Nousiainen (2015, 27) listaa keskisimmiksi organisaation liittyviksi ongelmiksi käyttöönotosta seuraavat organisaatiomuutokset, puutteet muutosjohtamisessa, johdon puutteellisen sitoutumisen projektiin, projektiryhmän ja sidosryhmien välisen kommunikaation puutteen, kulttuuriset erot, sekä käyttäjien perehdytyksen. Näistä erityisesti projektiryhmän ja sidosryhmien välisen kommunikaation, sekä käyttäjien kouluttamisen tärkeys nousevat esille monessa eri tutkimuksessa.

4 Toteutus

Toteutustapaa valittaessa oli tarkoitus luoda mahdollisimman paljon järjestelmän todellista käyttöympäristöä vastaavat olosuhteet. Järjestelmän käytettävyyttä on tarkasteltu kolmesta näkökulmasta asennusprosessiin ja purchase to pay sekä order to cash -prosesseihin perustuen, joista kaksi jälkimmäistä ovat liiketoiminnan perusprosesseja. Niiden aikana järjestelmissä suoritettavat toiminnot mahdollistavat järjestelmän käytettävyyden laajamittaisen arvioinnin aikaisemmissa tutkimuksissa tehtyjä havaintoja hyödyntäen.

Purchase to pay ja order to cash -prosessit vaativat onnistuakseen tarvittavien taustatietojen luomisen. Tutkimuksen aikana taustatietoina luotiin järjestelmää käyttävä kuvitteellinen yritys, toimittaja, asiakas ja tuote, joiden luominen vaatii järjestelmän eri moduuleihin perehtymistä ja niiden käyttämistä. Taustatietojen luominen ja näiden kahden prosessin suorittaminen vaatii kattavan käytettävyysarvion kannalta riittävästi perustoiminnallisuuksien käytön toistamista ja järjestelmän eri moduuleihin tutustumista. Lisäksi kaikki taustatietoihin ja prosesseihin liittyvät toiminnot ovat sellaisia, joita järjestelmän käyttäjät käyttävät päivittäisessä työssään.

4.1 Asennus virtuaaliympäristöön

Järjestelmien testaamista varten luotiin virtuaaliympäristö, johon kuului Windows 10 -työasema ja kolme palvelinta. Kullekin palvelimelle asennettiin yksi toiminnanohjausjärjestelmä ja järjestelmiä käytettiin joko työasemalle asennetulla asiakasohjelmalla tai selaimella http-yhteyden kautta, mikäli järjestelmä tarjosi verkkokäyttöliittymän. Taulukkoon 3 on listattu käytetyt palvelinkäyttöjärjestelmät ja asiakasohjelmat järjestelmäkohtaisesti.

Taulukko 3. Palvelinkäyttöjärjestelmät ja asiakassovellukset järjestelmittäin

Toiminnanohjausjärjestelmä	Palvelimen käyttöjärjestelmä	Asiakasohjelma
Metasfresh	Xubuntu 16.04. LTS	Verkkokäyttöliittymä Google Chrome -selaimella
Tryton	Xubuntu 16.04 LTS	Tryton Desktop Client 5.4
Axelor	Windows Server 2019 Desktop Experience	Verkkokäyttöliittymä Google Chrome -selaimella

4.2 Tutkimuksen tarkastelun kohteet

Järjestelmiä tarkasteltiin kolmesta näkökulmasta, joiden avulla voitiin tehdä johtopäätöksiä niiden toiminnasta ja käytettävyydestä. Tarkastelun kohteina olivat järjestelmän asennusvaihe ja konfigurointi, peruskäytettävyys ja ulkoasu, sekä saatavilla oleva käyttäjätuki ja ohjeistus.

Asennusvaiheen ja tarvittavien konfiguraatioiden tekemisen haastavuutta arvioitiin niiden helppouteen, vaatimaan aikaan ja kustomointimahdollisuuksiin perustuen. Nämä ovat tekijöitä, joita joissakin tutkimuksissa on käytetty ohjelmiston asennettavuuden mittaamisessa. (Lauras, M. ym. 2015. 68; Lenhard, J. 2016 125-127.)

Järjestelmien käytettävyyttä on tarkasteltu teoriataustassa mainituissa tutkimuksissa tehtyihin havaintoihin ja niissäkin huomioituun Nielsenin (1994) kymmenen heuristisen säännön listaan perustuen. Käytettävyyttä on arvioitu sellaisen käyttäjän näkökulmasta, jolla on kokemusta toiminnanohjausjärjestelmän käyttämisestä, mutta jolle testattavat järjestelmät ovat uusia.

Käyttäjätuki ja ohjeet luetaan yleisesti osaksi käytettävyyttä, mutta ne huomioitiin erillisenä osa-alueena niiden suuren merkityksen vuoksi. Aiemmissa tutkimuksissa on havaittu, että vieraskieliset käyttöohjeet voivat aiheuttaa joillekin käyttäjille hankaluuksia. Myös käyttäjäkoulutus koettiin kyselytutkimuksen mukaan tärkeänä. (Järvelä 2014, 40; Mikkonen 2012, 44.)

5 Testatut toiminnanohjausjärjestelmät

5.1 Metasfresh

Metasfresh on Saksassa sijaitsevan metas GmbH:n vuodesta 2006 alkaen kehittämä GPL v3 -lisenssillä julkaistu toiminnanjärjestelmä, jonka asentaminen ja käyttö omalla palvelimella on täysin ilmaista. Metasfresh perustuu Java ja JavaScript -ohjelmointikieliin. (Github 2020a; Metasfresh 2020a.)

Metasfresh sisältää valmiiksi asennettua kattavasti erilaisia moduuleita, joita ovat asiakkuuksienhallinta, myynti, osto, valmistuksenhallinta, varastonhallinta, toimitusketjunhallinta, laskutus, maksujenhallinta, kirjanpito, hinnastot, laadunhallinta ja sopimustenhallinta. (Metasfresh 2020f).

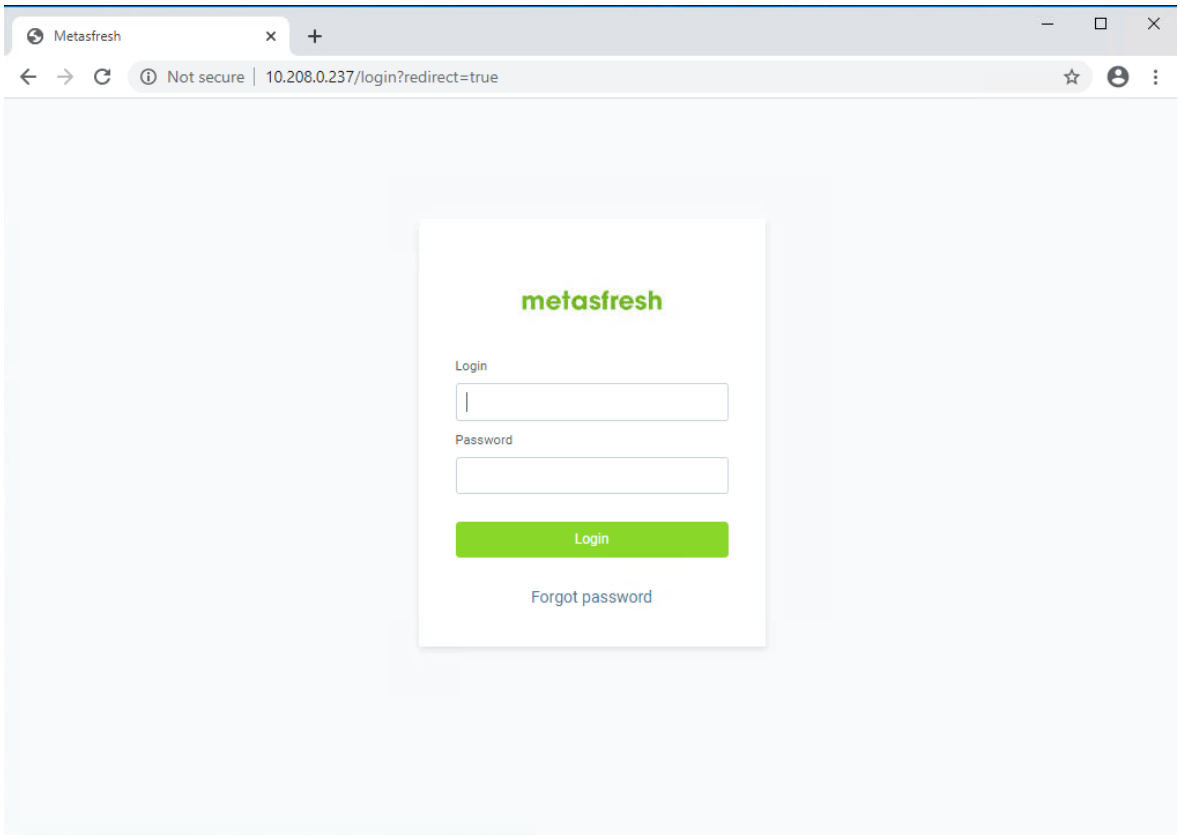
5.1.1 Asennusvaihe ja konfigurointi

Kun tämä tutkimus tehtiin, uusin Xubuntun versio, jota Metasfresh tuki oli 16.04 LTS, joten asennus tehtiin kyseiseen käyttöjärjestelmäversioon (Metasfresh 2020b). Palvelimelle oli asennettu Xubuntun graafisella käyttöliittymällä varustettu versio, mutta järjestelmän asennus ja konfiguraatiot tehtiin komentokehotteen kautta.

Asentaminen oli Linux-palvelimet tuntevalle vaivatonta, kaikki tarvittava oli ladattavissa Xubuntun pakettienhallinnan kautta, käyttäjän täytyi ajaa yksi asennustiedosto, avata tarvittavat portit tulimuuriin ja tehdä tarvittavat konfiguraatiot verkko-osoitteiden osalta, minkä jälkeen järjestelmän verkkokäyttöliittymä oli valmis käytettäväksi (Metasfresh 2020c). Web-käyttöliittymän lisäksi on tarjolla myös asiakassovellus, joka vaatii toimiakseen Javan asentamisen (Metasfresh 2020d).

Asennusohjeen (Metasfresh 2020c) mukaisesti suoritetun asennusvaiheen aikana ei ilmennyt ongelmia. Asennusprosessia voi kuvailla helpoksi, sillä käytännössä kuka tahansa Linuxia joskus käyttänyt pystyisi asennusohjeen avulla suorittamaan onnistuneen asennuksen. Asennusvaiheeseen ei kulu juurikaan aikaa, järjestelmä oli käyttövalmis noin viidentoista minuutin kuluttua asennuksen aloittamisesta. Kustomoitavuutta ei asennuksen osalta juurikaan tarjottu, kaikki moduulit ja tietokanta asentuivat asennustiedoston ajamisen yhteydessä.

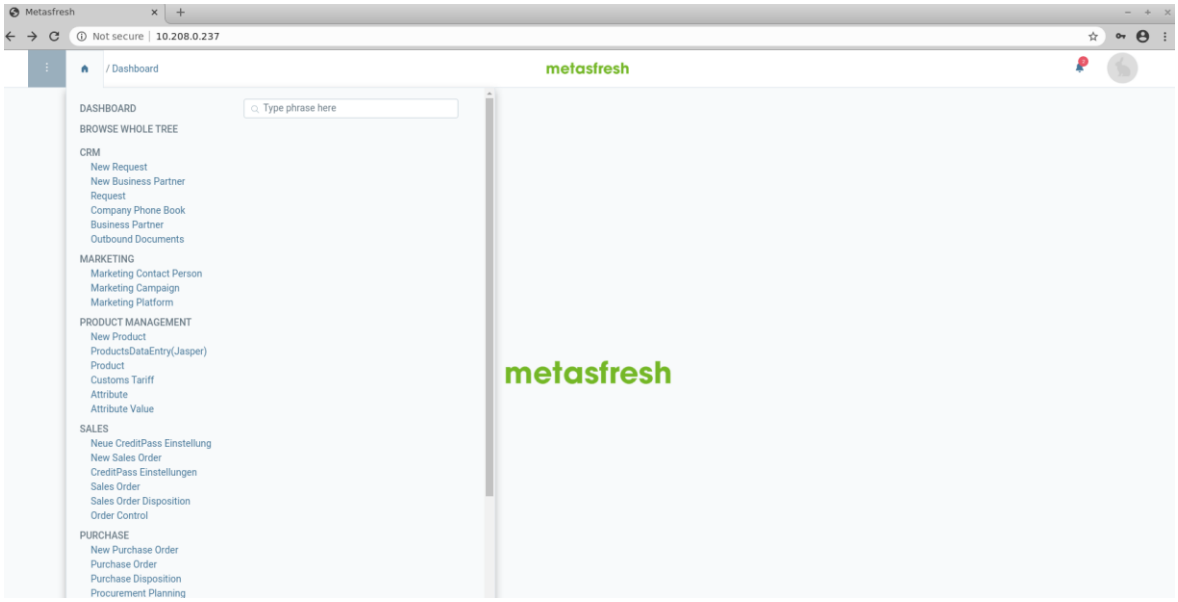
Näiden huomioiden perusteella Metasfreshin asennettavuuden voidaan arvioida olevan hyvä, kustomointimahdollisuuksia lukuun ottamatta.



Kuva 4. verkkokäyttöliittymän kirjautumissivu

Verkkokäyttöliittymään kirjaudutaan ensimmäisen kerran oletuskäyttäjätunnuksella, jonka oikeuksin käyttäjä voi salasanan vaihtamisen jälkeen luoda oman käyttäjän ja avata järjestelmään oman yrityksensä. Tämän jälkeen järjestelmä on valmis kirjanpidon tilien, tuotteiden ja kumppaneiden luomista varten. Kuvassa 4 näkyy verkkokäyttöliittymän kirjautumissivu.

5.1.2 Peruskäytettävyys ja ulkoasu

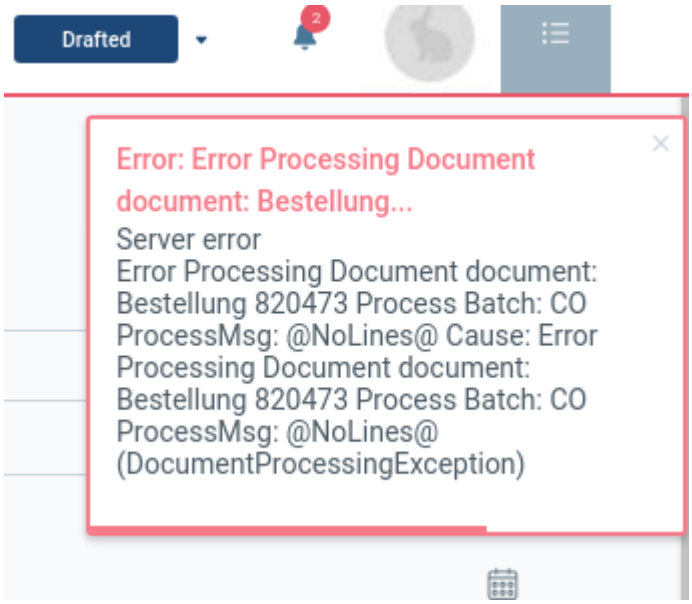


Kuva 5. Käyttäjän pääsivu pikavalikon ollessa avattuna

Metasfresh on ulkoasultaan selkeä ja sitä voi kuvailla kohtalaisen modernin näköiseksi, värimaailma on pirteä, mutta maltillinen. Kaikki oletusmoduulit asentuvat automaattisesti, joten päävalikosta löytyy paljon toimintoja. Eri toimintojen välillä siirtymistä ja niiden löytämistä helpottamaan on tehty hakukenttä, joka täyttää tarkoituksensa hyvin. Ilman hakukenttää eri toimintojen etsiminen pitkältä listalta veisi paljon aikaa. Kuvassa 5 näkyy Metasfreshin etusivu, johon on avattu pikavalikko, jossa näkyy osa moduuleista ja niiden toiminnoista.

Moduulit on otsikoitu selkeästi ja kaikki niiden toiminnot löytyvät suoraan listalta, joten yksittäisen toiminnon avaaminen ei vaadi käyttäjältä kyseisen moduulin avaamista erikseen. Toiminnot on otsikoitu niin, että ne löytyvät helposti myös hakukentän avulla.

Tarjolla olevan dokumentaation avulla järjestelmän peruskäytön oppii nopeasti ja toiminnot, kuten esimerkiksi myynti- ja ostotilaus jakavat saman ulkoasun, sekä toimivat samalla taustalogiikalla, mikä nopeuttaa järjestelmän käytön oppimista, mikäli käyttäjä käyttää työssään useampaa eri moduulia. Mikkonen (2011, 45) on käytettävyystutkimuksensa johtopäätöksissä suositellut sovellusten käyttöliittymien yhdenmukaistamista, mikä täyttyy Metasfreshin kohdalla.



Kuva 6. Virheilmoitus

Kun käyttäjä tekee virheen, antaa järjestelmä kuvan 6 mukaisen virheilmoituksen. Kuvan tapauksessa yritettiin luoda ostotilaus lisäämättä tuoterivejä. Ilmoituksesta tulee ilmi, mistä virhe johtuu, mutta siinä on paljon käyttäjän kannalta turhaa tekstiä, mikä vaikeuttaa sen tulkitsemista. Lisäksi ilmoitus on näkyvillä vain tietyn aikaa, jonka jälkeen sen saa näkyviin uudelleen vain toistamalla virheen. Kyseinen virheilmoitus on monilta osin Nielsenin (1994) virheilmoituksia koskevan heuristisen säännön vastainen, sillä sen ei voida katsoa olevan selkeä, eikä yksiselitteisesti antavan ilmi virheen syytä ja ehdottavan ratkaisua siihen.

Järjestelmän antamat ilmoitukset ja niiden sisältö on huomioitu Nielsenin (1994) listalla kahdessa eri kohdassa, listalla todetaan olevan tärkeää, että järjestelmä kommunikoi aktiivisesti käyttäjän kanssa erilaisin ilmoituksin, minkä voidaan katsoa Metasfreshin kohdalla tapahtuvan, mutta on myös tärkeää, että ilmoitukset ovat selkeitä, yksiselitteisiä ja ehdottavat käyttäjälle ratkaisua, mikä ei Metasfreshin virheilmoitusten kohdalla toteudu.

Järjestelmä ei tutkimushetkellä tukenut suomen kieltä, mikä voi olla joillekin käyttäjille ongelmallista, kuten Järvelä (2014, 40) oli havainnut tutkimuksessaan käyttöohjeistuksen ja käyttöliittämän kielen osalta.

5.1.3 Käyttäjätuki ja ohjeet

Metasfreshin kotisivuilta (Metasfresh 2020e) löytyy kattavasti dokumentaatiota ja ohjeita järjestelmän eri toiminnoista, sekä niiden käytöstä. Monessa ohjeessa on tekstin lisäksi erillinen animaatio, jossa ohjeen aiheena oleva toiminto suoritetaan. Tämä voidaan nähdä

käyttöönoton kannalta positiivisena asiana, sillä käyttäjäkoulutuksen lisäksi kattavat työohjeet helpottavat uuden järjestelmän käytön oppimista. Ilmaista käyttäjätukea tarjotaan Metasfreshin virallisella keskustelupalstalla (Metasfresh 2020e), josta löytyy hakukoneen avulla vastaus yleisimpiin kysymyksiin.

Metasfreshin dokumentaatio on saatavilla vain englannin- ja saksankielisenä.

5.2 Tryton

Tryton on sen taustalla toimivan yhdistyksen kehittämä toiminnanohjausjärjestelmä. Sen ensimmäinen versio on julkaistu vuonna 2008. Tryton on julkaistu GPLv3 -lisenssillä ja sen perusmoduulit ovat käyttäjän omalle palvelimelle asennettuna maksuttomia.

Tryton on toteutettu Python ja JavaScript -ohjelmointikielillä (Github 2020b). Tryton perustuu toiseen avoimen lähdekoodin toiminnanohjausjärjestelmään nimeltä OpenErp (Lwm 2008), jonka myöhemmät versiot tunnetaan nimellä Oodoo (Oodoo 2020).

Trytoniin asennettavissa olevia kehitystiimin ylläpitämiä moduuleita ovat kirjanpito, myynti, varastonhallinta, asiakkuuksienhallinta, osto, toimitusketjunhallinta, tuotannonhallinta, toimitustenhallinta, projektinhallinta ja laskutuksenhallinta. (Tryton 2020a).

5.2.1 Asennusvaihe ja konfigurointi

Tryton tukee useita Linux-pohjaisia käyttöjärjestelmiä (Tryton 2020b), tässä tutkimuksessa käyttöjärjestelmäksi valittiin Xubuntu 16.04. LTS. Palvelimelle oli asennettu Xubuntun graafisen käyttöliittymän sisältävä työpöytäversio, mutta järjestelmän asennus ja siihen liittyvät konfiguraatiot tehtiin komentokehotteen kautta.

Ennen Trytonin asentamista palvelimelle täytyi asentaa Python-pakettienhallinta nimeltä pip, jonka kautta itse järjestelmä ja sen toimintaa varten tarvittavat ohjelmat asennetaan. Pakettienhallinnan lisäksi täytyy asentaa PostgreSQL -tietokannanhallintajärjestelmä.

Tietokannanhallintajärjestelmän asentamisen jälkeen täytyi siihen tehdä pieniä konfiguraatioita ja luoda tietokanta Trytonia varten. Tryton ei sisällä valmiiksi yhtäkään moduulia, joten ne kaikki asennetaan erikseen, moduuleita voi asentaa joko Xubuntun oman tai Python-pakettienhallinnan kautta. Asentamisen jälkeen moduulit täytyi päivittää tietokantaan erillisellä komennolla, jotta ne tulivat näkyviin Trytonissa.

Asennusvaiheessa vei eniten aikaa moduulien asentaminen ja niiden päivittäminen tietokantaan. Tässä vaiheessa korostui Markus Savolaisen (2008, 53) tutkimuksessaan mainitsema järjestelmien perustuntemuksen tärkeys. Linux-järjestelmän tuntemisesta hakemistorakenteiden ja vähintään peruskomentojen osalta on merkittävää hyötyä niin tiedon etsimisessä Internetistä, kuin asennusvaiheeseen liittyvien ongelmien ratkaisussa.

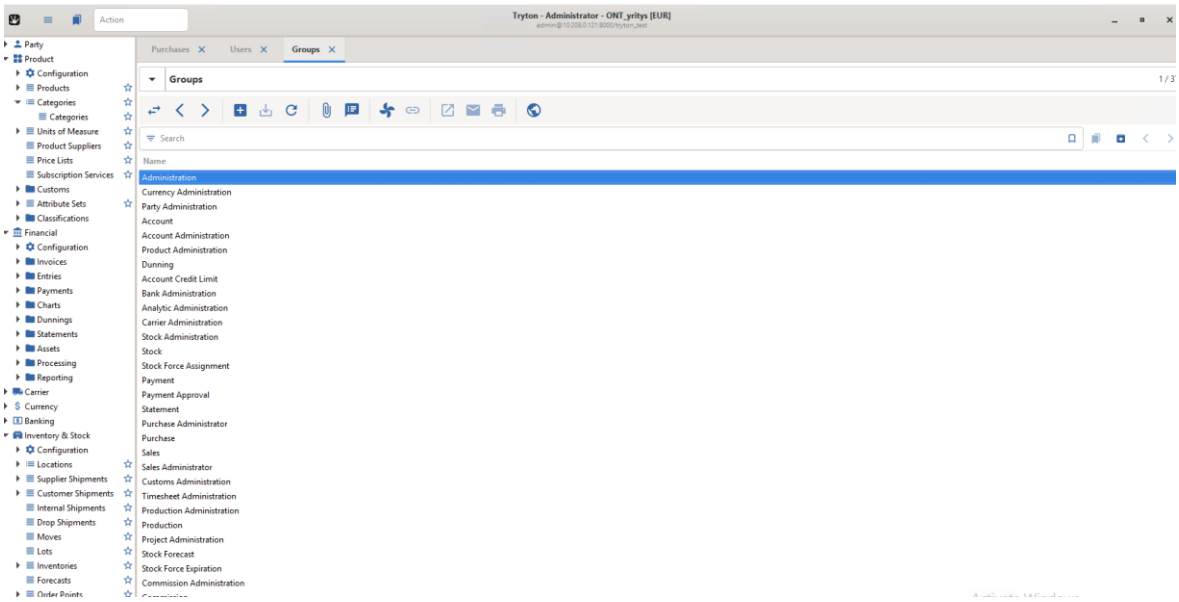
Asennuksen jälkeen täytyi konfiguroida tietokantayhteys, verkko-osoitteet ja avata tarvittavat portit tulimuuriin. Tämän jälkeen järjestelmään pääsi kirjautumaan asiakasohjelman kautta konfiguraatiotiedostoon määritellyllä verkko-osoitteella. Palvelimelle asennetut moduulit täytyi asentaa vielä erikseen järjestelmän graafisen käyttöliittymän kautta, jotta ne sai käyttöön.

Trytonin asentaminen vaati taustatyötä tietokannan ja Python-pakettienhallinnan osalta, lisäksi käyttäjän on luotava tietokanta itse. Asennukseen ei myöskään ollut tarjolla kaikkia asennusvaiheita kattavaa virallista ohjeistusta, joten se on monilta osin ensikertalaiselle työläs ja aikaa vievä prosessi. Moduulien asentaminen erikseen ensin palvelimelle ja sitten vielä erikseen Trytoniin graafisen käyttöliittymän kautta tuo asennusprosessiin kaksi ylimääräistä työvaihetta. Koko prosessin suorittamiseen kului tiedonhaku ja moduulien asennus mukaan lukien useita tunteja. Asennuksen kustomoitavuus on erikseen asennettavien moduulien ansiosta hyvä, koska käyttäjällä on vapaus valita haluamansa moduulit. Kokonaisuutena asennusprosessia voidaan kuvata työlääksi.

Kuva 7. Trytonin asiakasohjelma, johon on syötetty kirjautumistiedot

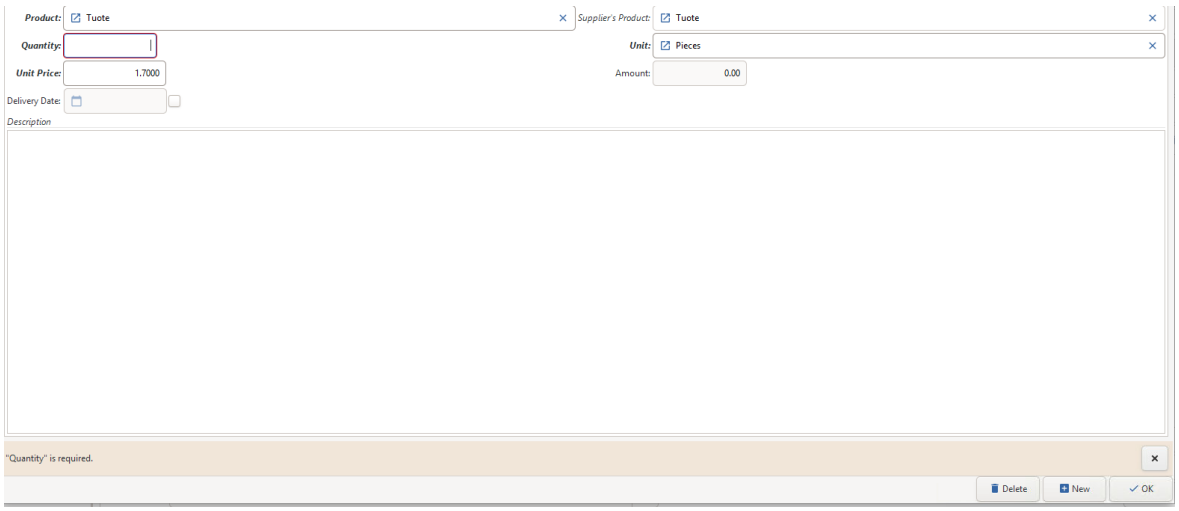
Asiakasohjelma tallensi kirjautumistiedot onnistuneen kirjautumisen jälkeen, joten käyttäjän ei tarvitse huolehtia kuin siitä, että käyttäjänimi ja salasana ovat oikein, käyttäjän salasanaa kysytään vasta connect -painikkeen napauttamisen jälkeen, kun yhteys on muodostunut onnistuneesti. Kuvassa 7 asiakassovellus ja siihen testivaiheen ensimmäisen kirjautumisen aikana syötetyt tiedot.

5.2.2 Peruskäytettävyys ja ulkoasu

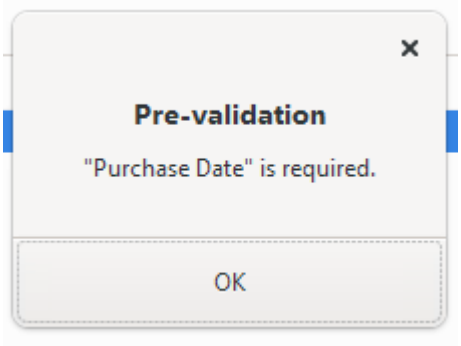


Kuva 8. Trytonin käyttöliittymä

Kuvassa 8 näkyvän Trytonin käyttöliittymän oletusulkoasu on vanhahtava. Peruskäytettävyydeltään se on kuitenkin selkeä, eri moduulit löytyvät nopeasti ikkunan vasemmassa reunassa olevasta navigointipalkista ja niiden välillä siirtymistä voi nopeuttaa ikkunan yläosaan sijoitettua hakukenttää käyttämällä. Käyttöliittymän selkeyden vuoksi peruskäytön oppi nopeasti, ikkunat ja toiminnot noudattavat samaa peruslogiikkaa, joten perustoiminnot, kuten uuden rivin lisääminen tapahtuvat aina samasta paikasta, mikä helpottaa muiden moduulien oppimista yksittäisen moduulin käyttämisen jälkeen. Mikkosen (2011, 45) huomautus toimintojen yhdenmukaisuudesta toteutuu Trytonin kohdalla.



Kuva 9. ”Quantity is required” -virheilmoitus ostotilausta luodessa



Kuva 10. Virheilmoitus ponnahdusikkunassa

Trytonin virheilmoitukset täyttävät Nielsenin (1994) listan asettamat kriteerit, niitä voi kuvailla ytimekkäiksi ja tietyissä tapauksissa puutteellisten kenttien kohdalla käytetään korostusvärejä, kuten kuvan 9 tapauksessa. Puutteellisten tietojen täyttäminen ja virheilmoituksia aiheuttaneiden tekijöiden korjaaminen oli järjestelmän käyttöä aloittaessa helppoa, sillä niissä kerrottiin suoraan virheen syy. Kuvan 9 kaltaisen virheilmoituksen lisäksi Tryton antaa joissakin tapauksissa myös kuvan 10 mukaisia ponnahdusikkunalla toteutettuja virheilmoituksia.

Tryton ei tue suomen kieltä, joten kaikilta käyttäjiltä vaaditaan vähintään perustason englannin kielen osaaminen.

5.2.3 Käyttäjätuki ja ohjeet

Trytonin kotisivuilta (Tryton 2020c) löytyvä eri moduuleihin keskittyvä dokumentaatio avaa eri moduuleita ja niiden toimintaa yleisluontoisesti, eikä sen voi katsoa riittävän käyttöohjeeksi. Käyttöohjeiden puute johtaa siihen, että käyttäjän on opeteltava useita monimutkaisiakin asioita itse. Testausvaiheessa merkittävimpänä tällaisena asiana tuli ilmi kirjanpidon tilien ja niiden keskinäisten riippuvuuksien opettelu kokeilemalla. Aiemmin käsitellyistä selkeistä virheilmoituksista oli apua tässä vaiheessa.

Trytonin kotisivun hae apua -linkki (Tryton 2020a) johtaa viralliselle foorumille, josta löytyy vastaus yleisimpiin kysymyksiin ja tutkimuksen aikana tehtyjen havaintojen perusteella kehitystiimin jäsenet lukevat ja vastailevat kysymyksiin melko aktiivisesti, joten apua on saatavilla kohtuullisella odotusajalla.

Dokumentaatiota tai foorumikeskustelua ei ole tarjolla suomeksi.

5.3 Axelor

Axelor on AGPLv3 -lisenssillä julkaistu avoimen lähdekoodin toiminnanohjausjärjestelmä, jota kehittää ranskalainen Axelor SAS (Axelor 2020a).

Axelor on toteutettu Java ja JavaScript -ohjelmointikielillä (Github 2020b). Sen käyttö perusmoduuleineen on käyttäjän omalle palvelimelle asennettuna maksutonta.

Axeloriin on tarjolla toiminnanohjausjärjestelmien perusmoduulit, kuten myynti, markkinointi, osto, varastonhallinta, tuotannonhallinta, laadunhallinta, kirjanpito, laskutus ja budjetointi. Perusmoduulien lisäksi Axeloriin on saatavilla moduuli myös henkilöstönhallintaan, henkilöstönhallinnan moduulista löytyy toiminnot esimerkiksi kaluston ja kustannustenhallintaa varten. (Axelor 2020e)

5.3.1 Asennusvaihe ja konfigurointi

Järjestelmä asennettiin Windows Server 2019 Database Desktop Experience -palvelimelle, jossa oli graafinen käyttöliittymä. Järjestelmän asennustavaksi valittiin ladattavan WAR-paketin avulla tehtävä asennus.

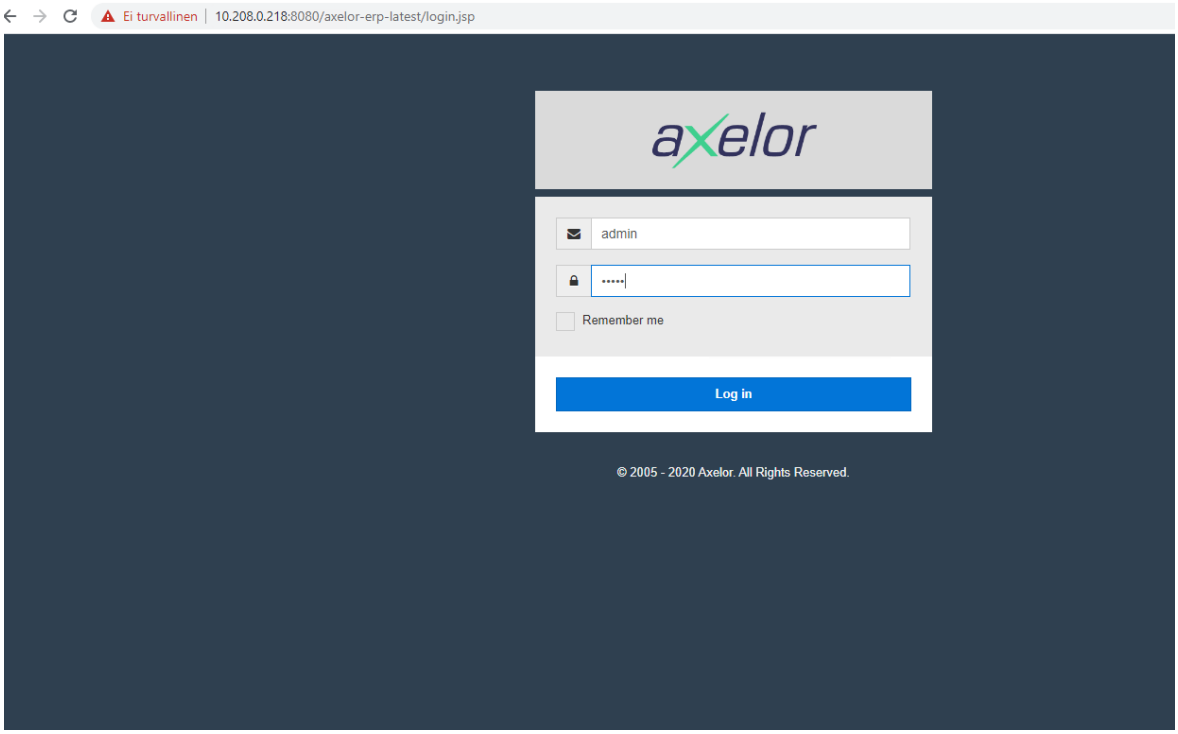
Axelor vaatii toimiakseen seuraavat kolmansien osapuolien ohjelmat: OpenJDK 8, Tomcat 8.5 ja PostgreSQL:n version 9.4 tai sitä uudemman. Asennusohjeeseen (Axelor 2020b) on listattu osoitteet kaikkien ohjelmien lataamiseen niiden ylläpitäjien virallisilta sivuilta. Kyseisiltä sivuilta löytyi myös asennusohjeet erilaisia asennustapoja varten.

Axelor vaatii toimiakseen PostgreSQL-tietokannan, joka käyttäjän on luotava itse. Tietokannan luomiseen ei ollut erillistä ohjetta, joten siihen täytyi etsiä erilliset ohjeet. PostgreSQL tarjoaa sivuillaan (PostgreSQL 2020) kattavan dokumentaation ohjelman käyttöä varten. Tietokannan luominen graafisen käyttöliittymän kautta tuotti aluksi hankaluuksia, mutta dokumentaation perehtymisen jälkeen se onnistui ongelmitta.

Taustaohjelmien asentamisen ja tietokannan luomisen jälkeen ladattiin ja purettiin Axelorin sisältävä WAR-paketti hakemistoon C://Tomcat8/webapps/. Paketin mukana tulleeseen konfiguraatiotiedostoon tehtiin tarvittavat muutokset asennusohjeen mukaisesti, minkä jälkeen Tomcatin voi käynnistää. Ensimmäinen käynnistyskerta vei useita minuutteja, sillä järjestelmä loi sen aikana toimintaa varten tarvittavan sisällön aiemmin luotuun tietokantaan.

Tietokantayhteyteen, järjestelmän käyttämiin tallennushakemistoihin ja verkkokäyttöliittymän osoitteeseen vaadittavat konfiguraatiot tehdään palvelimella sijaitsevaan application.properties -tiedostoon. Tarvittavien asetusten tekeminen onnistui helposti Axelorin ohjeita (Axelor 2020b) seuraamalla.

Axelorin asennus tapahtuu pääosin automaattisesti, kun se käynnistetään ensimmäisen kerran. Eniten työtä vaati asennuksen valmistelu, jonka yhteydessä asennetaan Axelorin toimintaa varten vaaditut kolmansien osapuolien ohjelmat. Asennusprosessi vei useita tunteja, josta suuri osa kului Tomcat 8.5:n ja PostgreSQL:n käyttöön tutustumiseen, tiedonhakuun ja konfigurointiin. Asennuksen kustomoitavuus on hyvä, sillä käyttäjä voi valita haluamansa moduulit, jotka asennetaan graafisen käyttöliittymän kautta Axelorin asentamisen jälkeen. Kokonaisuutena Axelorin asennettavuuden voidaan sanoa olevan melko hyvä, varsinkin jos käyttäjällä on kokemusta Tomcatista ja PostgreSQL:stä, muussa tapauksessa asennusvaihe vaatii jonkin verran lisätyötä niihin perehtymisen osalta.



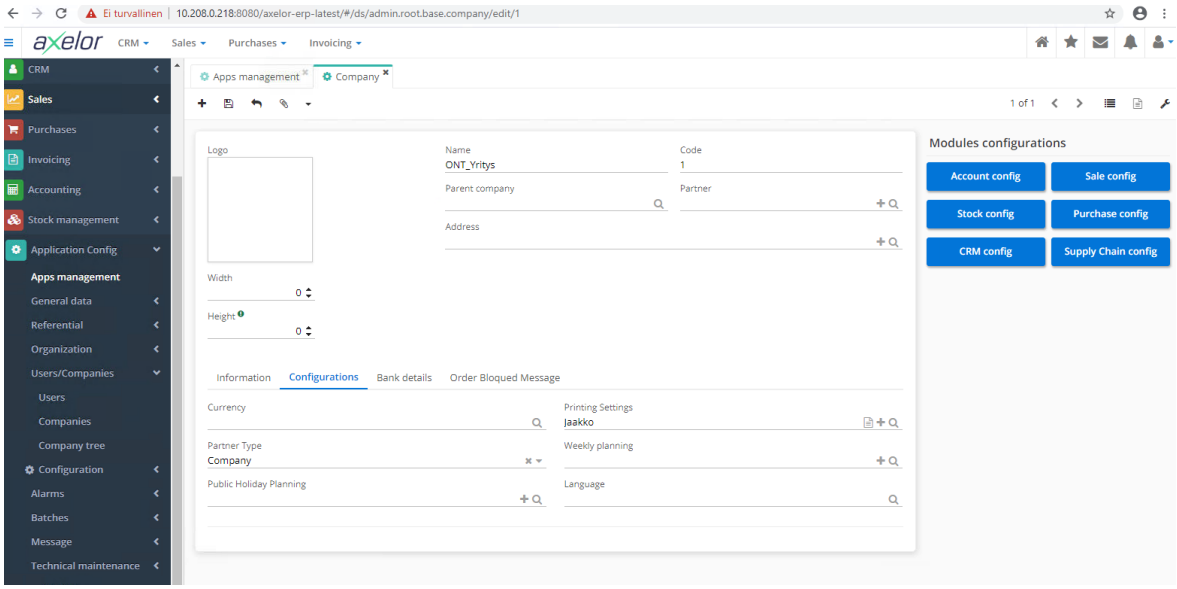
Kuva 11. Axelorin verkkokäyttöliittymän kirjautumissivu

Kun Tomcat on käynnistynyt, avautuu Axelorille määritellyssä http-osoitteessa kuvassa 11 näkyvä kirjautumissivu.

Axelorin moduulien asentaminen ja konfigurointi tehdään graafisessa käyttöliittymässä ja siihen löytyy jonkin verran ohjeistusta Axelorin käyttödokumentaatiosta. Käytännössä moduulit asennetaan yksitellen ja konfiguroidaan sen jälkeen, mikäli jokin moduuli vaatii toi-

miakseen toisen vielä asentamattoman moduulin, järjestelmä asentaa vaaditut moduulit automaattisesti.

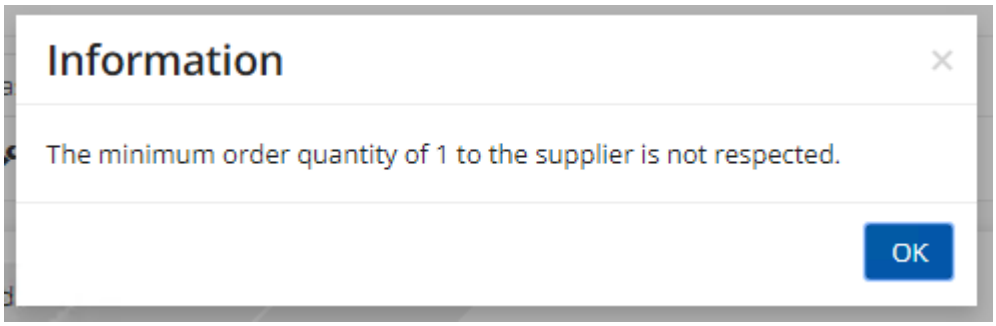
5.3.2 Peruskäytettävyys ja ulkoasu



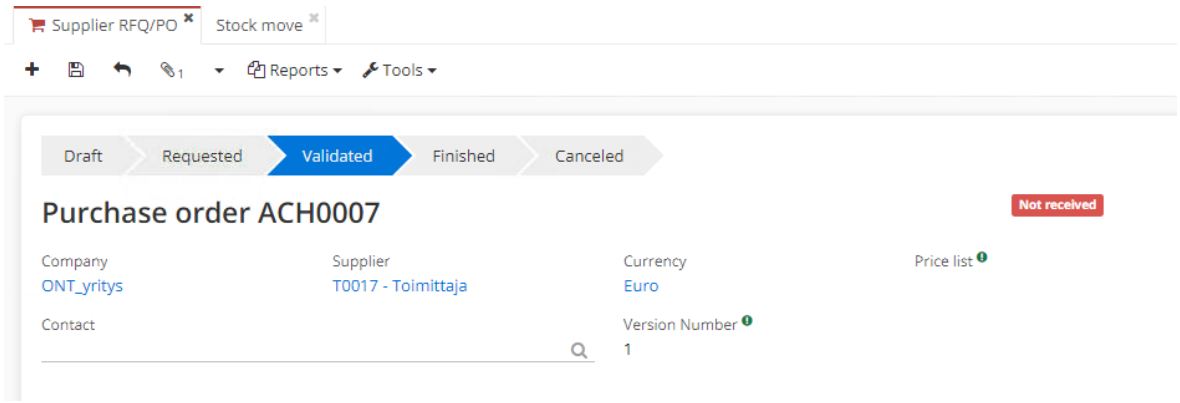
Kuva 12. Axelorin käyttöliittymä, johon on avattu ikkuna yrityksen luomista varten

Kuvassa 12 näkyvä peruskäyttöliittymä tuntui heti alussa selkeältä ja eri toiminnot oli helppo löytää ytimekkäästi otsikoitujen moduulien alta. Peruskäyttöä häiritsee eniten toimintoihin liittyvän hakukentän puute, mikä pakottaa käyttäjän siirtymään eri toimintojen välillä päävalikon kautta, hakukenttä säästäisi aikaa varsinkin järjestelmää jo jonkin aikaa käyttäneen käyttäjän kohdalla, joka muistaa eri toimintojen nimet. Tältä osin järjestelmä ei täytä Nielsenin (Nielsen 1994) heuristisen sääntöjen listan käytön joustavuutta ja tehokuutta käsittelevää kohtaa.

Muista testatuista järjestelmistä poiketen Axelor vaatii muutamaa poikkeusta lukuun ottamatta tehtyjen muutosten tallentamisen erillistä tallennuspainiketta käyttämällä. Mikäli jonkin ikkunan yrittää sulkea ennen tallentamista, järjestelmä ilmoittaa, että tallennetut tiedot menetetään ja esittää haluatko jatkaa, muttei anna vaihtoehtoa tallentaa ikkunan sulkemisen yhteydessä.



Kuva 13. Axelorin virheilmoitus



Kuva 14. Tilausikkunan työvaiheet

Virheilmoitukset ovat selkeitä, eivätkä ne jätä tulkinnanvaraa, kuten kuvasta 13 ilmenee. Käyttäjä tietää ilmoituksen luettuaan, mistä virhe johtui. Eri toimintojen onnistumisesta ei tule erillistä ilmoitusta, mutta se on helppo havaita ikkunassa tapahtuvista muutoksista. Kuten kuvassa 14 nähdään, osto- ja myyntitilausten ikkunan yläosaan on sijoitettu työjono, josta tilauksen työvaihe ja sen muuttuminen on nähtävillä helposti, sillä meneillään oleva vaihe on korostettu.

Axeloria ei saa suomenkielisenä, joten muiden testattujen järjestelmien tavoin käyttäjiltä vaaditaan englannin kielen taitoa.

5.3.3 Käyttäjätuki ja ohjeet

Axelorin kotisivuilla (Axelor 2020c) on tarjolla kattavasti dokumentaatiota eri moduuleista ja niiden käytöstä englannin kielellä. Dokumentaatiossa on käytetty paljon kuvakaappauksia ja polut, joista toiminnot löytyvät on kuvattu selkeästi eri vaiheiden alussa. Aivan perustason käyttöohjeistus, jossa toimintojen käyttäminen käydään läpi vaiheittain kuitenkin puuttuu.

Järjestelmän kotisivuilta löytyy myös virallinen keskustelufoorumi (Axelor 2020d), jolla on keskustelua eri ongelmiin ja ominaisuuksiin liittyen. Tutkimushetkellä foorumi vaikutti kui-

tenkin melko hiljaiselta, etusivulla oli useampi kysymys odottamassa vastausta, ja joidenkin vastauksetta jääneiden kysymysten julkaisusta oli ehtinyt kulumaan useampi päivä. Viestejä selaamalla löytyi kuitenkin paljon viestiketjuja, joissa kehitystiimin jäsenet olivat osallistuneet keskusteluun, joten kehitystiimi seurasi foorumia, muttei erityisen aktiivisesti.

5.4 Keskeiset havainnot

Taulukossa 4 kuvataan käyttötestausvaiheen aikana tehdyt keskeisimmät havainnot.

Taulukko 4. Käyttötestausvaiheen keskeiset havainnot

Järjestelmä	Asennusvaihe	Peruskäytettävyys ja ulkoasu	Käyttäjätuki ja ohjeet
Metasfresh	Linux-järjestelmien perusteet tuntevalle helppo ja nopea asentaa. Asennettavuus on kokonaisuutena hyvä.	Valikot ja ikkunat ovat selkeitä, peruskäyttö on nopeasti opittavissa. Virheilmoitukset ovat sekavia ja jättävät tavalliselle käyttäjälle paljon tulkinnanvaraa.	Tarjolla olevat käyttöohjeet ovat yksityiskohtaisia ja erilaista dokumentaatiota on tarjolla kattavasti.
Tryton	Asentaminen vaatii syvempää perehtyneisyyttä Linux-palvelimiin ja erilaisten ohjelmien asentamiseen. Myös tietokantaosaaminen on tarpeen. Asennettavuus on kokonaisuutena monivaiheinen ja vaativa.	Peruskäyttöliittymä on vanhahtava, mutta selkeä ja toimiva. Perustoimintojen käytön oppii nopeasti. Järjestelmän antamat ilmoitukset ovat yksiselitteisiä, eivätkä ne jätä tulkinnanvaraa.	Aivan perustason ohjetta, jossa eri prosessien vaiheet olisi kuvattu vaiheittain ei ole tarjolla järjestelmän käytön tai asentamisen osalta.
Axelor	Asentaminen Windows-palvelimelle vaatii Windows-järjestelmien perustuntemusta ja ymmärrystä tietokannoista. Asennettavuus on kokonaisuutena melko hyvä, mutta käyttäjältä vaaditaan Tomcat:in ja PostgreSQL:ään perehtyneisyyttä.	Järjestelmän perustoiminnot ovat nopeasti opittavissa ja ilmoitukset ovat yksiselitteisiä. Käyttöliittymän suurin yksittäinen puute on pikahaun puuttuminen.	Eri toimintoihin löytyy kuvakaappauksin varustettuja yksityiskohtaisia ohjeita, mutta aivan perustason ohjeistus puuttuu. Ilmaista käyttötukea tarjoava foorumi vaikutti tutkimushetkellä hiljaiselta.

6 Pohdinta

6.1 Johtopäätökset

Järjestelmätoimittaja on käyttöönottoprojektin kannalta merkittävä osapuoli, joka jää kokonaan pois, mikäli yritys tai organisaatio päättää ottaa avoimen lähdekoodin toiminnanohjausjärjestelmän käyttöönsä omin resurssein. Tällöin jo projektin suunnitteluvaiheessa on hyvä huomioida Henri Teittisen kuvaus järjestelmäimplementaation kolmesta maailmasta ja Riku Niemisen tutkimuksessa havaitut yleisimmät käyttöönoton ongelmat, jotta perustason virheiltä vältyttäisiin. (Nieminen 2018, 27; Teittinen 2008, 172.)

Tämän tutkimuksen aikana ilmeni, että Markus Savolaisen (2008, 53) yleisluontoinen huomio projektiryhmän riittävästä järjestelmäosaamisesta on merkittävä ja järjestelmätoimittajan puuttuessa sen tärkeys korostuu erityisesti riittävän toiminnanohjausjärjestelmien toiminnan tuntemisen osalta.

Tuukka Paanasen (2017, 28) havaintojen perusteella projektiryhmän tulee ymmärtää myös liiketoimintaa, jotta käyttöönottoprojekti takaisi järjestelmän hyödyntämisen kaiken sen tarjoaman potentiaalin osalta. Tämän huomion tärkeyttä voidaan myös tässä tutkimuksessa tehtyjen havaintojen perusteella korostaa. Järjestelmän yhteensopivuus käyttäjäyrityksen omien liiketoimintaprosessien ja toimintatapojen kanssa on suositeltavaa pyrkiä huomioimaan jo ennen järjestelmän lopullista valintaa.

Järjestelmien käytettävyyden kannalta merkittävä yksittäinen tutkimuksen aikana esille noussut asia on järjestelmän käyttöliittymän ja tarjolla olevan dokumentaation kieli. Tätä tutkimusta tehtäessä suomen kieli ei ollut mukana yhdenkään tutkitun järjestelmän kielivalikoimassa. Kielen merkitys käytettävyyteen on huomioitu Nielsenin kymmenen heuristisen säännön listalla, lisäksi se on nostettu esiin myös Järvelän käytettävyystutkimuksessa. (Järvelä 2014, 40; Nielsen 1994.)

Edellä mainittujen käytettävyydestä tehtyjen havaintojen perusteella voidaan suositella suomenkielisten peruskäyttöohjeiden luomista ennen käyttöönottovaihetta, mikäli kaikki järjestelmän käyttäjät eivät koe kielitaitonsa olevan riittävällä tasolla, jotta he pärjäisivät tarjolla olevilla vieraskielisillä ohjeilla. Käyttäjien kielitaito on hyvä huomioida myös käyttöliittymän kielen osalta jo toiminnanohjausjärjestelmää valittaessa.

Toinen esille noussut asia on käyttötuki, joka jää tarjolla olevan dokumentaation ja virallisten keskustelupalstojen varaan. Vaikka kaikkien testattujen järjestelmien osalta vastauksia

yleisimpiin kysymyksiin niin järjestelmän toiminnasta kuin asennusvaiheesta löytyi kattavasti, ei vapaaehtoisuuteen perustuvilla keskustelupalstoilla ole sovittuja vasteaikoja tai sopimuksen takaamaa takuuta mahdollisen ongelman aktiivisesta selvittämisestä.

Nielsenin (1994) kymmenen heuristisen säännön viimeinen kohta käsittelee dokumentaatiota ja sen saatavuutta painottaen, että tuen ja ohjeiden tulee olla helposti saatavilla ja niissä tulee käydä ohjeistettu asia läpi vaiheittain. Tutkimus osoitti, että Metasfresh oli tutkituista järjestelmistä ainoa, johon valmiiksi tarjolla ollut käyttöohjeistus täytti Nielsenin asettamat kriteerit. Tämän havainnon perusteella on suositeltavaa varautua ohjeiden kääntämisen lisäksi luomaan eri työvaiheista omia yksityiskohtaisia käyttöohjeita.

Järjestelmien peruskäytettävyyden voidaan tutkimushavaintojen perusteella katsoa olevan kunnossa, mutta mikään niistä ei täytä kaikkia Nielsenin (Nielsen 1994) heuristisen sääntöjen listan kymmenestä arviointikriteeristä. Merkittävimpinä puutteina nousivat esille Metasfreshin epäselvät virheilmoitukset ja Axelorin hakukentän puute, jotka ovat molemmat pieniä, mutta osaltaan järjestelmän käyttötehokuuteen vaikuttavia asioita.

6.2 Yhteenveto

Tutkimuksen aikana tehtyihin havaintoihin perustuen avoimen lähdekoodin toiminnanohjausjärjestelmät tarjoavat perustoiminnan osalta saman kuin suljetun lähdekoodin järjestelmät. Käytettävyyden ja perusominaisuuksien osalta avoimen lähdekoodin järjestelmä voi parhaimmillaan sopia pienen ja keskisuuren yrityksen tai yhteisön käyttöön erittäin hyvin.

Tutkimuksen perusteella avoimen lähdekoodin järjestelmän suurimpana ongelmana nousee esiin riittävän käyttäjäkoulutuksen ja -tuen järjestäminen. Suljetun lähdekoodin järjestelmien käyttäjäkoulutuksista ja -tuesta muodostuu kustannuksia, mutta puutteellinen käyttäjäkoulutus ja tuki aiheuttavat niin ikään kustannuksia esimerkiksi menetetyn työajan osalta. Ongelmat käytön ja käytettävyyden suhteen voivat pahimmillaan pysäyttää kokonaisia toimitusketjuja ja tulla yritykselle kalliiksi, kuten Mikko Sahanen (2014, 63) huomauttaa käytettävyystutkimuksensa yhteenvedossa.

Toinen merkittävä ero suljetun lähdekoodin järjestelmien hyväksi on järjestelmätoimittajan myötä tuleva räätälöintimahdollisuus. Mikäli käyttöönottoa harkitsevassa yrityksessä ei ole valmiiksi resursseja avoimen lähdekoodin järjestelmän kehittämiseen, tulee osaavan työntekijän palkkaamisesta tai palvelun ostamisesta ulkopuoliselta taholta kustannuksia. Tutkimuksessa testattujen avoimen lähdekoodin järjestelmien lisenssit ovat kuitenkin tapauk-

sesta riippumatta ilmaisia, joten vaikka palvelintilan, kehityksen ja ylläpidon ostaisi ulkopuoliselta taholta, jää lisenssien tuoma kustannuserä siitä huolimatta pois.

Ulkopuolisten tahojen osaamisen hyödyntämistä järjestelmän räätälöimisessä omaan käyttöön sopivaksi kannattaa harkita, sillä kuten Joni Kettunen (2018, 35) on tutkimuksessaan todennut, niin avoimen lähdekoodin toiminnanohjausjärjestelmät eivät ole riippuvaisia tietyistä järjestelmätoimittajista ja kehittäjästä, vaan niihin voi tehdä muutoksia mikä tahansa taho, joten tarjouksia räätälöimisestä voi pyytää vapaasti miltä tahansa ohjelmistoalan toimijalta.

Projektiryhmällä, jolla on riittävä asiantuntemus, ja joka perehtyy ennen käyttöönottoprojektin aloittamista huolellisesti järjestelmään ja sen toimintaan, kiinnittäen huomiota kattavaan testaukseen, loppukäyttäjien kouluttamiseen ja järjestelmän sopivuuteen osaksi omia liiketoimintaprosesseja, on hyvät mahdollisuudet onnistuneeseen avoimen lähdekoodin toiminnanohjausjärjestelmän käyttöönottoon. Riittävä asiantuntemus tarkoittaa tässä tapauksessa käytettävän palvelinkäyttöjärjestelmän tuntemista, kokemusta tietokannoista ja niiden hallinnasta, aiempaa kokemusta toiminnanohjausjärjestelmän käyttöönotosta ja yritysten tietoverkoista, sekä valitun toiminnanohjausjärjestelmän ohjelmointikielen tuntemista. Teknisen osaamisen lisäksi on tärkeää, että projektiryhmässä on edustettuna liiketoiminnan eri osa-alueita.

Tässä tutkimuksessa tehtyjen havaintojen perusteella avoimen lähdekoodin toiminnanohjausjärjestelmä voisi sopia parhaiten sellaisen pienen yrityksen käyttöön, jolla on riittävät resurssit ylläpitää ja päivittää järjestelmää itse, ja jolle järjestelmä tarjoaa riittävät ominaisuudet ilman jatkokehitystä. Tällaisessa tapauksessa lisenssimaksujen ja ulkopuolisen järjestelmätoimittajan kanssa toteutetun käyttöönottoprojektin kustannusten poisjäämisen voidaan arvioida tuovan merkittävää taloudellista hyötyä yrityksen sisäisesti suorittaman käyttöönoton tuomiin kustannuksiin nähden. Räätälöintipotentiaalia ei kuitenkaan pidä jättää huomioimatta, mikäli sille on tarvetta, eikä mitään järjestelmää kannata poissulkea ennen räätälöinnin kustannusten selvittämistä.

Tämän tutkimuksen perusteella Metasfresh voidaan arvioida kolmesta vertaillusta järjestelmästä varteenotettavimmaksi vaihtoehdoksi pienen yrityksen toiminnanohjausjärjestelmänä. Kokonaisuutena Metasfresh ja sen käyttöön, sekä asennukseen tarjolla oleva dokumentaatio tarjoavat vertailluista järjestelmistä parhaat puitteet käyttöönottoprojektin onnistumiselle kustannustehokkaasti, aina asennusvaiheesta loppukäyttäjien koulutukseen ja varsinaiseen käyttöönottoon.

Kaikilla tutkituilla järjestelmillä oli omat hyvät ja huonot puolensa. Yksikään niistä ei ollut täydellinen, eikä yksikään osoittautunut merkittävästi muita huonommaksi vaihtoehdoksi. Vaikka Metasfresh tarjoaa kokonaisuutena parhaat mahdollisuudet käyttöönoton onnistumiselle, ovat muutkin testatut järjestelmät toimivia vaihtoehtoja, kunhan käyttöönottoprojektin suunnitteluvaiheessa huomioidaan tutkimuksessa ilmi tulleet asiat.

6.3 Tulosten luotettavuus ja hyödynnettävyys

Järjestelmien käytettävyyden osalta tutkimusta voidaan pitää luotettavana, koska havainnot perustuvat useampaan aiemmin tehtyyn tutkimukseen, joissa moni havainto on toistunut eri tutkimuskohteissa.

Vaikka tutkimuksen pohdinnassa valittiin kolmesta tutkitusta järjestelmästä tämän tutkimuksen perusteella pienen yrityksen käyttöön parhaiten sopiva järjestelmä, ei sen paremmuudesta muihin tutkittuihin järjestelmiin verrattuna voida pelkästään tämän tutkimuksen perusteella tehdä lopullisia johtopäätöksiä. Järjestelmän sovittaminen olemassa oleviin prosesseihin on organisaatiokohtaista, eikä tässä tutkimuksessa huomioitu käyttöönottovaihetta kuin teoriatasolla.

Tutkimuksen tuloksia voidaan käyttää apuna, kun harkitaan avoimen lähdekoodin toiminnanohjausjärjestelmää vaihtoehtona yrityksen tai yhteisön toiminnanohjausjärjestelmäksi. Tästä tutkimuksesta on erityisesti hyötyä, mikäli harkittava järjestelmä on jokin tutkimuksen aikana testatuista järjestelmistä, sillä järjestelmien arvioimisen lisäksi niiden asennusvaiheessa syntyneestä dokumentaatiosta on apua niitä asennettaessa.

6.4 Opinnäytetyöprosessin ja oman oppimisen arviointi

Tutkimusprosessi tarjosi mielenkiintoisen katsauksen avoimen lähdekoodin toiminnanohjausjärjestelmien maailmaan, mikä ei ollut itselleni tutkimuksen alkuvaiheessa lainkaan tuttu. Minulla oli työhistoriani myötä kokemusta erilaisista suljetun lähdekoodin toiminnanohjausjärjestelmistä ja niiden käyttöönottoprojektista, mutta toiminnanohjausjärjestelmän asentaminen ja käytettävyyden arvioiminen teoriatasolla olivat minulle uusia asioita.

Prosessin edetessä pääsin hyödyntämään opintojen aikana erityisesti palvelinkäyttöjärjestelmistä kertynyttä osaamista ja oppimaan sen myötä paljon uutta myös varsinaisen tutkimuksen ulkopuolelle jääneistä asioista. Toiminnanohjausjärjestelmien osalta pääsin tutustumaan niiden toimintaan pintaa syvemmin, mikä helpotti graafisen käyttöliittymän toimintojen taustalla tapahtuvien asiakokonaisuuksien ymmärtämistä.

Tutkimuksen teoriataustaan tutustumisen kautta löysin paljon hyviä todellisten järjestelmien toimintaan perustuvia havaintoja ja näkemyksiä, joista osa toistui useammassa tutkimuksessa, mikä vahvistaa niiden luotettavuutta.

Opinnäytetyöprosessi onnistui mielestäni hyvin, melko tiukasta aikataulusta huolimatta onnistuin joitakin pieniä yksittäisiä tehtäviä lukuun ottamatta pysymään aikataulussa ja sain tehtyä kaikki suunnittelemani asiat. Syvempi teoriataustaan tutustuminen jäi työn kirjoitusvaiheeseen, vaikka siihen olisi ollut hyvä paneutua kunnolla jo järjestelmien testausvaiheen alussa, jolloin olisin voinut suunnitella testausvaiheen paljon tarkemmin ja se olisi palvellut paremmin työn kirjoittamisvaihetta ja teoriataustaa. Tällöin minun ei olisi tarvinnut tehdä pieniä täydentäviä lisätestejä järjestelmissä enää tutkimuksen kirjoittamisvaiheen aikana.

Lähteet

Axelor 2020a. Axelor-lisenssi. Luettavissa: https://www.axelor.com/licence/ Luettu 04.05.2020.

Axelor 2020b. Asennusohje. Luettavissa: https://docs.axelor.com/abs/5.0/install/war/windows.html Luettu: 04.05.2020.

Axelor 2020c. Axelorin dokumentaatio. Luettavissa: https://docs.axelor.com/ Luettu: 01.05.2020.

Axelor 2020d. Keskustelufoorumi. Luettavissa: https://forum.axelor.com/ Luettu: 11.05.2020.

Axelor 2020e. All Modules. Luettavissa: https://www.axelor.com/erp-open-source/integrated-modules/ Luettu 20.05.2020.

Bistasolutions 2018. The 9 phases of ERP implementation. Luettavissa: https://www.bistasolutions.com/resources/blogs/9-erp-implementation-phases/ Luettu 18.05.2020.

Coss 2020. Avoin lähdekoodi. Luettavissa: https://coss.fi/avoimuus/avoin-lahdekoodi/ Luettu: 12.05.2020.

Datavtech 2019. A Brief History of ERP Software. Luettavissa: https://www.datavtech.com/a-brief-history-of-erp-software/ Luettu: 05.05.2020.

Europa 2020. Yleinen tietosuoja-asetus. Luettavissa: https://europa.eu/youreurope/business/dealing-with-customers/data-protection/data-protection-gdpr/index_fi.htm Luettu: 19.05.2020.

Forum.metasfresh.org 2020. Metasfresh-keskustelufoorumi. Luettavissa: https://forum.metasfresh.org/ Luettu 03.05.2020.

Github 2020a. Metasfresh. Luettavissa: https://github.com/metasfresh Luettu: 20.05.2020.

Github 2020b. Tryton. Luettavissa: https://github.com/tryton Luettu: 20.05.2020.

Github 2020c. Axelor. Luettavissa: https://github.com/tryton Luettu 20.05.2020.

GNU 2020a. AGPLv2 -lisenssi. Luettavissa: http://www.gnu.org/licenses/agpl-3.0.html Luettu: 12.5.2020.

GNU 2020b. GPLv3 -lisenssi. Luettavissa: http://www.gnu.org/licenses/gpl-3.0.html Luettu 12.05.2020.

Hossain, L, Patrick, J & Rashid, M. 2002. The evolution of ERP Systems: A historical perspective. Luettavissa: https://faculty.biu.ac.il/~shnaidh/zooloo/nihul/evolution.pdf Luettu 17.05.2020.

Investopedia 2020a. Enterprise Resource Planning (ERP). Luettavissa: https://www.investopedia.com/terms/e/erp.asp Luettu 07.05.2020.

Investopedia 2020b. Purchase-to-pay. Luettavissa: https://www.investopedia.com/terms/p/purchasetopay.asp Luettu 12.05.2020.

Järvelä, S. 2014. Toiminnanohjausjärjestelmän käytettävyyden kehittäminen. Luettavissa: https://core.ac.uk/download/pdf/38063848.pdf Luettu: 07.05.2020.

Kettunen, J. 2018. Avoimen lähdekoodin toiminnanohjausjärjestelmät pk-yrityksissä. Luettavissa: https://lutpub.lut.fi/bitstream/handle/10024/155939/Kandidaatinty%c3%b6%20Kettunen%20lopullinen%20v1.pdf?sequence=1 Luettu 09.05.2020.

Lauras, M, Zelm, M, Archiméde, B, Bénaben, F & Doumeingts, G. 2015. Enterprise Interoperability: Interoperability for Agility, Resilience and Plasticity of Collaborations (I-ESA 14 Proceedings). Wiley-ISTE.

Lenhard, J. 2016. Portability of Process-Aware and Service-Oriented Software: Evidence and Metrics. Otto-Friedrich-Uni.

Leppioja, M & Tuomela, I 2017. Avoimen ja suljetun lähdekoodin data-analytiikkaohjelmistot kustannusnäkökulmasta. Luettavissa: https://lutpub.lut.fi/bitstream/handle/10024/135015/Kandidaatinty%C3%B6_Tuomela_Leppioja.pdf?sequence=2&isAllowed=y Luettu 16.05.2020.

Lwm 2008. Tryton ERP 1.0 released. Luettavissa: https://lwn.net/Articles/307653/ Luettu 20.05.2020.

Metasfresh 2020a. Perustietoa järjestelmästä. Luettavissa: https://metasfresh.com/en/team/ Luettu 04.5.2020.

Metasfresh 2020b. Järjestelmävaatimukset. Luettavissa: https://metasfresh.com/en/installing-metasfresh-ubuntu-server/ Luettu: 04.05.2020.

Metasfresh 2020c. Asennusohje Ubuntu-palvelimelle. Luettavissa: https://metasfresh.com/en/installing-metasfresh-ubuntu-server/ Luettu: 04.5.2020.

Metasfresh 2020d. Asiakassovellus. Luettavissa: https://metasfresh.com/en/client-installation/ Luettu 10.5.2020.

Metasfresh 2020e. Dokumentaatio. Luettavissa: https://metasfresh.com/en/contribute/#docs Luettu: 10.5.2020.

Metasfresh 2020f. Functional modules. Luettavissa: https://docs.metasfresh.org/webui_collection/EN/FunctionalModules.html Luettu 20.05.2020.

Mikkonen, K. 2011. Toiminnanohjausjärjestelmän käytettävyys – Asiakastyytyväisyyskysely. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/33478/Mikkonen_Kaisu.pdf?sequence=1&isAllowed=y Luettu: 04.05.2020.

Moog, S. 2018.
Pros & Cons of Open Source Software at the Enterprise Level Luettavissa: https://typo3.com/blog/pros-cons-of-open-source-software-at-the-enterprise-level?utm_medium=TYPO3%20Blog&utm_source=Blog%20Post%20-%20Pros%20%26%20cons%20of%20open-source%20software%20at%20the%20enterprise%20level&utm_campaign=Open%20Source%20CMS Luettu: 19.05.2020.

Nielsen, J. 1994. 10 Usability Heuristics for User Interface Design. Luettavissa: https://www.nngroup.com/articles/ten-usability-heuristics/ Luettu: 07.05.2020.

Nielsen, J. 2012. Usability 101: Introduction to Usability. Luettavissa: https://www.nngroup.com/articles/usability-101-introduction-to-usability/ Luettu: 07.05.2020.

Nousiainen, R. 2015. Toiminnanohjausjärjestelmän käyttöönoton ongelmat. Luettavissa: http://urn.fi/URN:NBN:fi:jyu-201503101451 Luettu: 15.05.2020.

Oodoo 2020. The Odoo Story. Luettavissa: https://www.odoo.com/blog/odoo-news-5/post/the-odoo-story-56 Luettu: 20.05.2020.

Paananen, T. 2017. ERP-järjestelmä implementoinnin onnistumistekijät. Luettavissa: https://aaltodoc.aalto.fi/bitstream/handle/123456789/27792/bachelor_Paananen_Tuukka_2017.pdf?sequence=1&isAllowed=y Luettu 05.05.2020

Postgresql 2020. Dokumentaatio. Luettavissa: https://www.postgresql.org/docs/ Luettu: 04.05.2020.

Sahanen, M. 2014. Toiminnanohjausjärjestelmä käyttäjän näkökulmasta: käytettävyyden kartoittamisen menetelmiä kohdeyrityksessä. Luettavissa: https://trepo.tuni.fi/bitstream/handle/10024/95229/GRADU-1398770130.pdf?sequence=1&isAllowed=y Luettu 04.05.2020.

Savolainen, M. 2008. Avoimeen lähdekoodiin pohjautuvan ERP-järjestelmän käyttöönotto-toprosessi. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/1124/Savolainen_Markus.pdf?sequence=1&isAllowed=y Luettu: 04.05.2020.

Sippola, T. 2017. Usability is a key element of user experience. Luettavissa: https://eu.landisgyr.com/better-tech/usability-is-a-key-element-of-user-experience Luettu: 18.05.2020.

Smith, A. 2017. Usability first – Why usability design matters to UI/UX designers Luettavissa: https://uxplanet.org/usability-first-why-usability-design-matters-to-ui-ux-designers-9dfb5580116a Luettu: 18.05.2020.

Stuart, G. 2019. The Pros and Cons of Open-source Tools. Luettavissa: https://orangematter.solarwinds.com/2019/02/15/the-pros-and-cons-of-open-source-tools/ Luettu 19.05.2020.

Techtarget 2020. Order to cash (OTC or O2C). Luettavissa: https://searcherp.techtarget.com/definition/order-to-cash-OTC-or-O2C Luettu 12.05.2020.

Tadviser 2020. ERP systems (world market). Luettavissa: http://tadviser.com/index.php/Article:ERP_systems_(world_market) Luettu 07.05.2020.

Teittinen, H. 2008. Näkymätön ERP. Luettavissa: https://jyx.jyu.fi/bitstream/handle/123456789/19204/9789513934354.pdf?sequence=1&isAllowed=y Luettu: 14.05.2020.

Tryton 2020a. Trytonin kotisivu. Luettavissa: https://www.tryton.org/ Luettu: 02.05.2020.

Tryton 2020b. Trytonin lataussivu. Luettavissa: https://www.tryton.org/download Luettu 10.05.2020.

Tryton 2020c. Dokumentaatio. Luettavissa: https://docs.tryton.org/en/latest/#modules Luettu: 11.5.2020.

Yrittäjät 2020. Yrittäjyys Suomessa. Luettavissa: https://www.yrittajat.fi/suomen-yrittajat/yrittajyys-suomessa-316363 Luettu: 16.5.2020.

Liitteet

Liite 1 Axerlor asennusdokumentaatio

Käyttöjärjestelmä: Windows Server 2019 Database Desktop Experience

Alkutoimet

Ennen itse Axelorin asentamista täytyy asentaa seuraavat ohjelmat:

- OpenJDK 8
- Tomcat 8.5 (Asennushetkellä 8.5.50 oli uusin toimiva, sitä uudemmat aiheuttivat virheitä järjestelmän toiminnassa. Kehittäjät olivat tietoisia ongelmasta.)
- PostgreSQL 9.4 (tai myöhempi versio)

OpenJDK 8

OpenJDK:n sai ladattua osoitteesta https://adoptopenjdk.net/, joka oli mainittu Axelorin asennusdokumentaatiossa osoitteessa https://docs.axelor.com/abs/5.0/install/war/windows.html. AdopOpenJDK -sivustolta löytyy asennusohje eri käyttöjärjestelmille, Windowsiin sekä graafiselle, että komentorivikäyttöliittymälle.

Tomcat 8.5

Tomcatin saa ladattua osoitteesta https://tomcat.apache.org/download-80.cgi, joka oli niin ikään mainittu Axelorin asennusdokumentaatiossa. Asennustiedoston lataamisen jälkeen asennus tapahtui graafisen käyttöliittymän kautta samaan tapaan kuin missä tahansa ohjelmassa.

Asennusessa kysytty Javan polku oli omalla kohdallani C:\Program Files\AdoptOpenJDK\jdk-8.0.252.09-hotspot\jre\bin. Käytännössä se löytyy AdoptOpenJDK:n asennushakemistosta.

Tomcat:iin kannattaa konfiguroida web-manager -toiminto, sillä sen kautta on helppo tarkistaa, onko jokin web-sivu toiminnassa ja tarvittaessa ottaa se toimintaan manuaalisesti.

Seurasin tätä ohjetta web-managerin käyttöönottamiseksi:
https://stackoverflow.com/questions/1321933/how-do-i-set-tomcat-manager-application-user-name-and-password-for-netbeans

PostgreSQL 9.4

Tarvittavan asennustiedoston sai ladattua Axelorin asennusdokumentaation mukaisesti osoitteesta https://www.postgresql.org/download/windows/. Asennus tapahtui ohjelmien tavoin graafisen käyttöliittymän kautta normaaliin tapaan.

WAR-paketti

Kun tarvittavat taustaohjelmat oli asennuttu, otin Axelorin sivuilta lataamani https://www.axelor.com/community/downloads/ War-paketin pystyy OpenJDK:n asentamisen myötä purkamaan komentokehotteen kautta ajamalla komennon *jar -xvf tiedostonnimi.war* hakemistossa, jossa se sijaitsee.
https://stackoverflow.com/questions/7882745/how-to-extract-war-files-in-java-zip-vs-jar

PgAdmin4 ja tietokannan luominen

Ohjeessa oli vinkki luoda järjestelmää varten axelor -niminen tietokanta Postgesql:n mukana asennetun PgAdmin4 -ohjelman avulla. Päätin tehdä niin.

Käynnistymisen jälkeen ohjelma pyysi antamaan master-salasanan. Salasanan antamisen jälkeen tietokontaa pääsi hallitsemaan graafisen käyttöliittymän kautta.

Tietokannan luomiseksi seurasin seuraavaa ohjetta:
https://www.postgresqltutorial.com/postgresql-create-database/

Konfigurointi

Seuraavaksi täytyy etsiä WEB-INF\classes\application.properties -tiedosto ja muikata sen sisältöä.

Alkuperäinen sisältö:

 # Database settings
 # ~~~~~

db.default.driver = org.postgresql.Driver

db.default.ddl = update

db.default.url = jdbc:postgresql://localhost:5432/axelor-open-suite

db.default.user = axelor

*db.default.password = ******

Sisältö muokattuna ohjeen mukaiseksi:

 # Database settings

 # ~~~~~

db.default.driver = org.postgresql.Driver

db.default.ddl = update

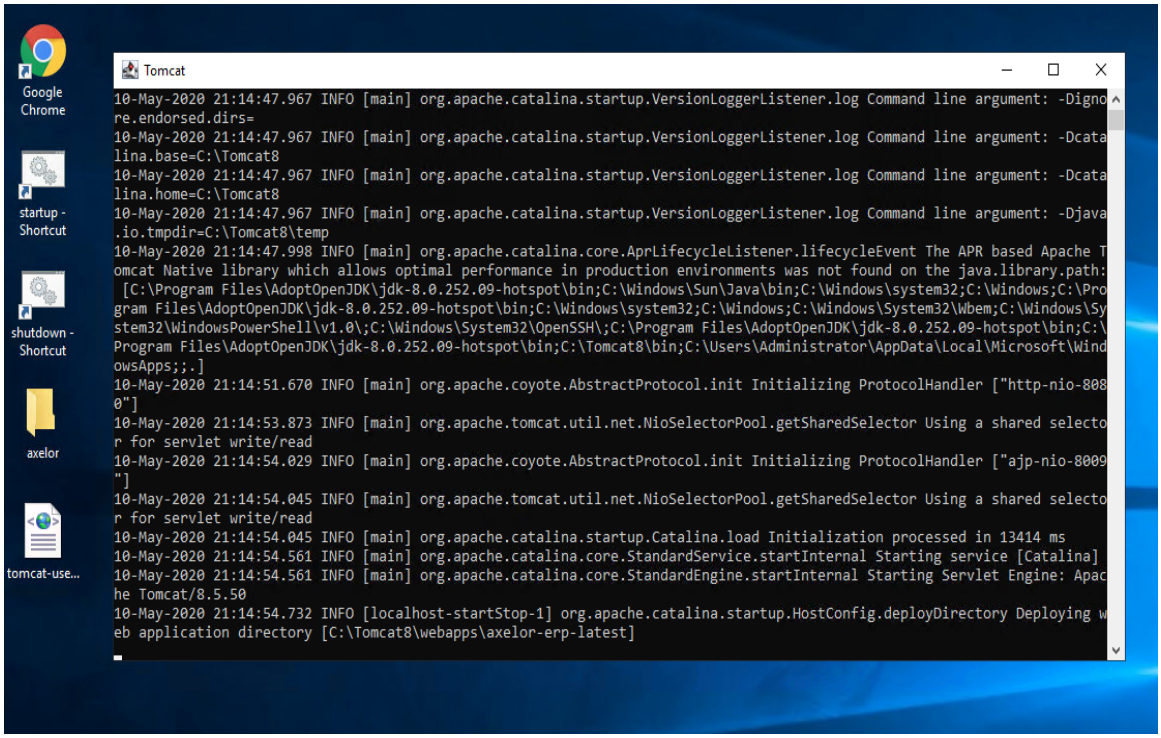
*db.default.url = jdbc:postgresql://localhost:5432/**axelor** (tietokannan nimi)*

*db.default.user = **postgres (tietokannan käyttäjätunnus)***

*db.default.password = **tietokannalle määritelty salasana***

Edellä tehtyjen toimenpiteiden jälkeen täytyi pysäyttää Tomcat, ellei se ole jo valmiiksi pysäytetty. Sen jälkeen Puretun war-paketin voi kaikkine hakemistoineen siirtää hakemistoon C:\Program Files\Apache Software Foundation\Tomcat 8.5\webapps. Kun hakemistokokonaisuus on siirretty, voi Tomcatin taas käynnistää.

Ensimmäinen käynnistys kestää jonkin aikaa, sillä Axelor luo tietokantaansa tarvittavan sisällön.



Kuva 1. Tomcatin käynnistysvaiheita komentotulkissa

Kun Tomcatin käynnistää sen asennushakemistosta löytyvästä startup-pikakuvakkeesta, pääsee käynnistyksen eri vaiheita ja mahdollisia virheilmoituksia seuraamaan komentotulkin kautta, kuten kuvassa 1 on tehty.

Liite 2 Tryton asennusdokumentaatio

Käyttöjärjestelmä: Xubuntu 16.04

Alkutoimet

Ohjelmiston voi asentaa Linuxin pip-pakettienhallinnan kautta. Aluksi täytyy kuitenkin asentaa pip-pakettienhallinta, ellei sitä ole jo valmiiksi asennettuna.

Asennus onnistuu komennolla:

sudo apt-get install python3-pip

Järjestelmän asennus

Järjestelmän asennus tapahtuu antamalla seuraavat komennot, järjestelmän lisäksi asennetaan muita tarvittavia paketteja.

sudo pip3 install trytond
sudo apt-get install libglib2.0-dev
sudo apt-get install libgirepository1.0-dev
sudo apt-get install python3-cairo
sudo apt-get install libcairo2-dev
pip3 install pycairo
pip3 install tryton

PostgreSQL:n asennus
Seuraavaksi asennetaan Postgresql -tietokanta, jonka Tryton vaatii toimiakseen.

Avataan pgdg.list -tiedosto:

sudo nano /etc/apt/sources.list.d/pgdg.list

Lisätään tiedostoon seuraava tekstinpätkä, jossa on asennushakemiston osoite, jotta asennus onnistuu pakettienhallinnan kautta:

“deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main”

Nyt Postgresql voidaan asentaa antamalla seuraavat komennot:

sudo apt-get update
sudo apt-get install postgresql postgresql-contrib

PostgreSQL:n konfigurointi

Avataan postgesql:n asetustiedosto komennolla:

sudo nano /etc/postgresql/11/main/postgresql.conf

Lisätään tiedostoon teksti:

listen_addresses = '*'

Käynnistetään postgesql uudelleen, jotta saadaan tehty muokkaus voimaan:

sudo systemctl restart postgresql

Sitten jatketaan komennoilla:

sudo ufw allow 5432/tcp
sudo apt-get install python-psycopg2
sudo apt-get install libpq-dev
sudo apt-get install python3-psycopg2

Seuraavaksi avataan tietokanta ja luodaan tietokanta Trytonia varten:

sudo -u postgres psql

Sitten luodaan tietokanta Trytonia varten komennoilla, tietokantakohtaiset osiot korostettu vihreällä värillä:

```
CREATE DATABASE tietokannan_nimi WITH OWNER = postgres ENCODING = 'UTF8'
LC_COLLATE = 'C' LC_CTYPE = 'C' TABLESPACE = pg_default CONNECTION LIMIT
= -1 TEMPLATE template0;
```

Ja jatketaan luomalla käyttäjä Trytonia varten:

```
CREATE ROLE käyttäjän_nimi WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE
INHERIT NOREPLICATION CONNECTION LIMIT -1 PASSWORD 'käyttäjän_salasana';
```

Lopuksi suljetaan tietokanta komennolla:

```
\q
```

Käyttöönotto

Luodaan hakemisto Trytond.config -tiedostolle ja itse tiedosto:

sudo mkdir /etc/tryton
sudo nano /etc/tryton/trytond.conf

Kopioidaan tiedostoon seuraava sisältö, laitekohtaiset muokattavat osiot korostettu vihreällä värillä:

```
# /etc/tryton/trytond.conf - Configuration file for Tryton Server (trytond)
 #
 # This file contains the most common settings for trytond (Defaults
 # are commented).
 # For more information read
 # /usr/share/doc/trytond-<version>/

 [database]
 # Database related settings

 # The URI to connect to the SQL database (following RFC-3986)
 # uri = database://username:password@host:port/
 # (Internal default: sqlite:// (i.e. a local SQLite database))
```

```
#
# PostgreSQL via Unix domain sockets
# (e.g. PostgreSQL database running on the same machine (localhost))
#uri = postgresql://tryton:tryton@/
#
#Default setting for a local postgres database
#uri = postgresql:///

#
# PostgreSQL via TCP/IP
# (e.g. connecting to a PostgreSQL database running on a remote machine or
# by means of md5 authentication. Needs PostgreSQL to be configured to accept
# those connections (pg_hba.conf).)
#uri = post-
gresql://tietokannan_käyttäjatunnus:tietokannan_salasana@localhost:5432/
uri = postgresql://
://tietokannan_käyttäjatunnus:://tietokannan_käyttäjatunnus@localhost:5432/

# The path to the directory where the Tryton Server stores files.
# The server must have write permissions to this directory.
# (Internal default: /var/lib/trytond)
path = /var/lib/tryton

# Shall available databases be listed in the client?
#list = True

# The number of retries of the Tryton Server when there are errors
# in a request to the database
#retry = 5

# The primary language, that is used to store entries in translatable
# fields into the database.
language = en
# en

[ssl]
# SSL settings
# Activation of SSL for all available protocols.
```

```
# Uncomment the following settings for key and certificate
# to enable SSL.

# The path to the private key
#privatekey = /etc/ssl/private/ssl-cert-snakeoil.key

# The path to the certificate
#certificate = /etc/ssl/certs/ssl-cert-snakeoil.pem

[jsonrpc]
# Settings for the JSON-RPC network interface

# The IP/host and port number of the interface
# (Internal default: localhost:8000)
#
# Listen on all interfaces (IPv4)

listen = 0.0.0.0:8000

#
# Listen on all interfaces (IPv4 and IPv6)
#listen = [::]:8000

# The hostname for this interface
#hostname =

# The root path to retrieve data for GET requests
#data = jsondata

[xmlrpc]
# Settings for the XML-RPC network interface

# The IP/host and port number of the interface
#listen = localhost:8069

[webdav]
# Settings for the WebDAV network interface
```

```
# The IP/host and port number of the interface
#listen = localhost:8080
listen = 0.0.0.0:8080

[session]
# Session settings

# The time (in seconds) until an inactive session expires
timeout = 3600

# The server administration password used by the client for
# the execution of database management tasks. It is encrypted
# using using the Unix crypt(3) routine. A password can be
# generated using the following command line (on one line):
# $ python -c 'import getpass,crypt,random,string; \
# print crypt.crypt(getpass.getpass(), \
# "".join(random.sample(string.ascii_letters + string.digits, 8)))'
# Example password with 'admin'
#super_pwd = jkUbZGvFNeugk
#super_pwd = <your pwd>


[email]
# Mail settings

# The URI to connect to the SMTP server.
# Available protocols are:
# - smtp: simple SMTP
# - smtp+tls: SMTP with STARTTLS
# - smtps: SMTP with SSL
#uri = smtp://localhost:25
uri = smtp://localhost:25

# The From address used by the Tryton Server to send emails.
from = tryton@<your-domain.tld>

[report]
# Report settings
```

```
# Unoconv parameters for connection to the unoconv service.
#unoconv = pipe,name=trytond;urp;StarOffice.ComponentContext

# Module settings
#
# Some modules are reading configuration parameters from this
# configuration file. These settings only apply when those modules
# are installed.
#
#[ldap_authentication]
# The URI to connect to the LDAP server.
#uri = ldap://host:port/dn?attributes?scope?filter?extensions
# A basic default URL could look like
#uri = ldap://localhost:389/

[web]
# Path for the web-frontend
#root = /usr/lib/node-modules/tryton-sao
listen = 0.0.0.0:8000
root = /usr/share/sao

[sale]
```

Konfigurointi

Konfigurointi tapahtuu asennuksen aikana edellisessä vaiheessa luodussa trytond.conf -tiedostossa. Ohjeita konfigurointiin löytyi osoitteesta: https://docs.tryton.org/projects/server/en/latest/topics/configuration.html

Kirjautuminen

user: admin

Pass: (salasana, jonka annoit tietokannalle asennusvaiheessa)

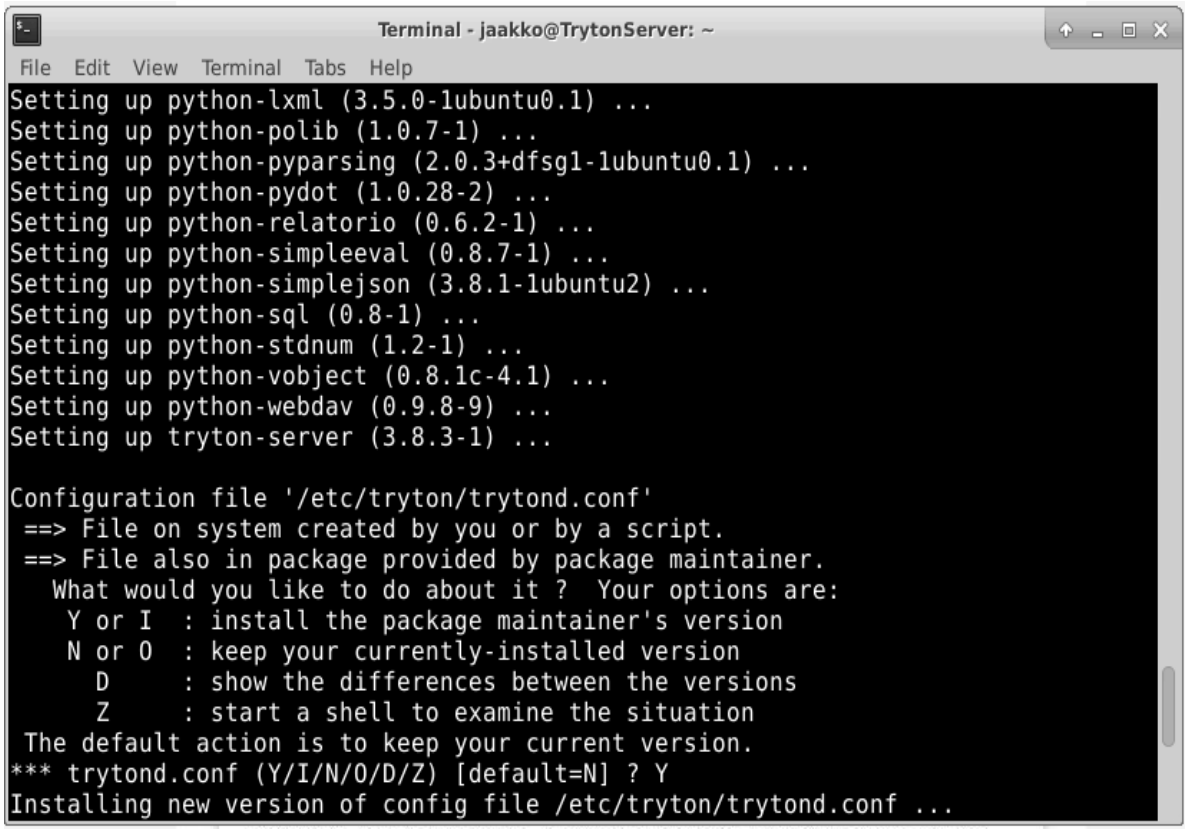
Moduulien asennus

Järjestelmän moduulit, kuten esimerkiksi myynti ja varastonhallinta asennetaan kaikki erikseen. Ubuntun kohdalla asennus hoituu helpoiten asentamalla kaikki moduulit pake-

tinhallinnan kautta, sillä tällöin käyttäjän ei tarvitse käydä jokaista moduulia läpi ennen asentamista. Kaikkien asennettujen moduulien käyttöönottaminen järjestelmässä ei ole välttämätöntä, joten ylimääräiset eivät häiritse peruskäyttöä.

Moduulien asennus tapahtuu antamalla seuraavat komennot:

Sudo apt-get update
Sudo apt-get install -y tryton-modules-all (-y antaa järjestelmälle luvan vastata kaikkiin asennusta koskeviin peruskysymyksiin automaattisesti kyllä, mikä säästää jonkin verran aikaa).



Kuva 1. Kysymys, halutaanko alkuperäinen konfiguraatiotiedosto korvata.

Asennuksen aikana järjestelmä kysyy, halutaanko aiemmin tämän dokumentaation aikana muokattu trytond.conf -tiedosto korvata paketinhallinnan kautta tulevalla, järjestelmän toiminnan kannalta tiedoston korvaaminen ei ollut pakollista, mutta sen voi myös halutessaan korvata, sillä alkuperäinen on helppo ottaa uudelleen käyttöön, koska järjestelmä tallentaa sen /etc/tryton/ -hakemistoon nimellä trytond.conf.old. Erona alkuperäiseen paketinhallinnan kautta tulevassa tiedostossa on oletuskonfiguraatiot muutamalle asennetulle moduulille.

Mikäli tiedoston korvaa paketinhallinnan versiolla, täytyy alkuperäiseen tehdyt muutokset tietokantayhteyden osalta kopioida siihen.

Kun moduulit on asennettu, täytyy ne vielä päivittää tietokantaan, jotta ne näkyvät Trytonissa, päivitys onnistuu komennolla:

*sudo trytond-admin -c /etc/tryton/trytond.com -d *tietokantasi_nimi* --update-modules-list*

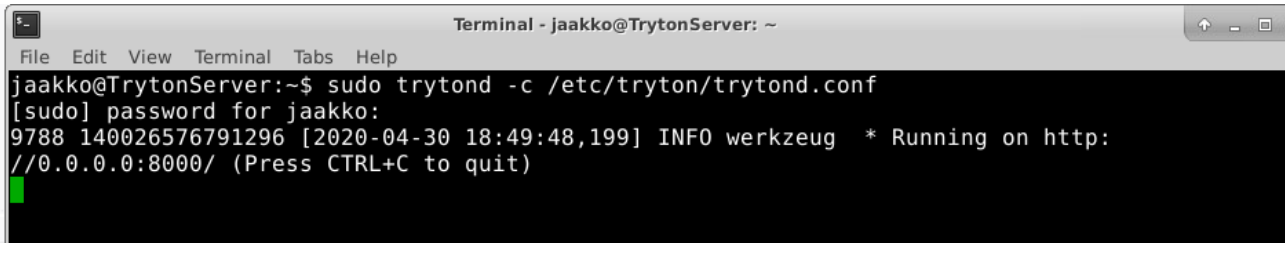
Tämän jälkeen Trytonin voi käynnistää komennolla:

sudo trytond -c /etc/tryton/trytond.conf

Kun kaikki on kunnossa, Tryton kertoo olevansa käynnissä osoitteessa http://0.0.0.0:8000/ Mikäli tässä kohdassa lukee 0.0.0.0:8000 sijaan Localhost:8000, on trytond.config tiedostossa väärä asetus [web] -kohdan alla, eikä palvelimeen saa yhteyttä ulkopuolisilta laitteilta.

Web-konfiguraatioiden tulisi siis näyttää tältä:

[web]
 # Path for the web-frontend
#root = /usr/lib/node-modules/tryton-sao
listen = 0.0.0.0:8000 (ei listen = localhost:8000)
root = /usr/share/sao



Tryton ei oletusasetuksilla käynnisty automaattisesti, vaan se täytyy käynnistää erikseen joka kerta, kun palvelin käynnistetään uudelleen.

Desktopt client

Järjestelmän käyttö tapahtuu Desktop clientin, eli asiakasohjelman kautta.

Asiakasohjelman saa ladattua Trytonin sivuilta erikseen, vaihtoehtona on joko viimeisin versio tai versio, johon tarjotaan pidennettyä tukea. Windowsille asentaminen tapahtuu .exe -tiedoston avulla, asennuksen jälkeen asiakassovellus on valmis käytettäväksi.

Valitsin aluksi pidennetyn tuen version, mutta yrittäessäni yhdistää palvelimelleni järjestelmä ilmoitti, että ” Incompatible version of the server”. Avasin Googlen selaimeen ja lähdin tutkimaan asiaa, pian selvisikin, että palvelimen ja asiakassovelluksen versioiden täytyy vastata toisiaan tietyssä määrin, jotta ne toimivat. Ns. LTS, eli pidennetyin tuen versio (Long Term Support) oli 5.0, kun palvelimelleni oli asennettu uusin versio, joka oli 5.4.6. Löytämäni ohjeen mukaan asiakasohjelman ja Trytonin versionumeroiden kahden ensimmäisen numeron tulee täsmätä, jotta ne toimivat keskenään.

Tarvittaessa Trytonin version voi tarkistaa palvelimelta komennolla:

trytond-admin –version

Lähteet

Asennus
https://discuss.tryton.org/t/installing-tryton-d-5-2-on-ubuntu-18-04-3-with-postgres-11-for-newbys-by-newby/1809

Versiokorjaus
https://stackoverflow.com/questions/13910356/tryton-incompatible-version-of-the-server

Client login
https://tryton-documentation.readthedocs.io/en/latest/user_guide/tryton_client.html

Tryton moduulit
https://docs.tryton.org/en/latest/

Moduulit ja komennot
https://pypi.org/search/?c=Framework+%3A%3A+Tryton

Kaikki moduulit

https://zoomadmin.com/HowToInstall/UbuntuPackage/tryton-modules-sale

Ohjeita perusasetuksiin

https://readthedocs.org/projects/tryton-administration-manual/downloads/pdf/latest/

Liite 3 Metasfresh asennusdokumentaatio

Käyttöjärjestelmä: Xubuntu 16.04

Alkutoimet

Asennustiedoston sai ladattua käyttäjän kotihakemistoon ohjelmiston kotisivulta: http://www.metasfresh.com/wp-content/releases/metasfresh-5_144.tar.gz

Tämän jälkeen asennustiedosto puretaan kotihakemistossa komennolla:

tar xvzf ./metasfresh-5_144.tar.gz

Purkamisen jälkeen siirrytään metasfresh_install -hakemistoon (**** tilalla komennossa järjestelmän käyttäjän nimi):

*cd /home/****/metasfresh_install*

Järjestelmän asennus

metasfresh_install -hakemistossa annetaan tiedostolle ensin suoritusoikeus ja aloitetaan asennus ajamalla se:

chmod +x ./install_metasfresh.sh
sudo ./install_metasfresh.sh

Käyttöönotto

Asennuksen jälkeen järjestelmä käynnistyi automaattisesti ja sen Web-käyttöliittymään pääsi syöttämällä selaimeen palvelimen IP-osoitteen. Palvelimen hallintasivulle, josta voi ladata työpöytäsovelluksen järjestelmän käyttämiseksi pääsee syöttämällä IP-osoitteen loppuun portin 8080, eli http://xx.xx.xx.xx:8080.

Metasfresh käynnistyy asennuksen jälkeen automaattisesti aina palvelimen käynnistyessä.

Järjestelmään voi kirjautua Web-käyttöliittymän kautta seuraavalla tunnuksella:

User: metasfresh
Password: metasfresh

Konfigurointi

Selaimen kautta toimivan webUI:n käyttöönotto vaatii asennuksen jälkeen käytännössä vain tarvittavien porttien sallimisen tulimuuriin ja tarvittaessa config.js -tiedoston muokkaamisen. Kyseiseen tiedostoon täytyy lisätä mahdollinen palvelimen domain-nimi.

Tässä tapauksessa domain-nimeä ei ole, koska kyse on tutkimuksesta, joten riitti, että kävin tarkistamassa asetusten olevan oikein.

Selain

Mikäli sivun avaa Firefox-selaimella, näkyy kirjautumiskentän alaosassa teksti ”Your browser might b enot fully supported. Please try Chrome in case of any errors”. Näin ollen siirryin käyttämään Google Chromea mahdollisten ongelmien välttämiseksi.

Config.js -tiedostoa pääsi muokkaamaan komennolla:

sudo nano /opt/metasfresh-webui-frontend/dist/config.js

Palvelimen toimintaa varten täytyy sallia seuraavat portit:

8080/TCP
61616/TCP
5432/TCP
80/TCP

Seuraavaksi seurasin webUI:n quick start -ohjetta vaihtaakseni oletuskäyttäjän salasanan. Seuraavaksi etsin ohjeen käyttäjän luomista varten ja loin sitä seuraten itselleni käyttäjän. Kun sain edellä mainitut vaiheet suoritetuksi, siirryin Quick Start -ohjeen vaiheeseen neljä, eli syötin järjestelmään kuvitteellisen yritykseni tiedot.

Quick start -ohjeen viidentenä vaiheena on esimerkkityönkulun läpikäynti, mikä vaikutti erittäin hyvältä järjestelmän käytön perusasioiden opettelussa, näin ollen suoritin vaiheen.

Quick Start -ohje

https://docs.metasfresh.org/webui_collection/EN/Quickstart.html

Käyttäjän luominen

https://docs.metasfresh.org/webui_collection/EN/New_system_user.html